

CFM 03110US
Appln. No. 10/600,813
CAU: NYA

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日 2 0 0 2 年 6 月 2 8 日
Date of Application:

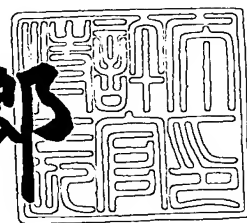
出 願 番 号 特 願 2 0 0 2 - 1 9 1 1 2 9
Application Number:
[ST. 10/C]: [J P 2 0 0 2 - 1 9 1 1 2 9]

出 願 人 キヤノン株式会社
Applicant(s):

2 0 0 3 年 7 月 1 0 日

特許庁長官
Commissioner,
Japan Patent Office

太田信一郎



出証番号 出証特 2 0 0 3 - 3 0 5 5 5 9 8

【書類名】 特許願

【整理番号】 4741005

【提出日】 平成14年 6月28日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 9/00

【発明の名称】 ログ取得方法およびプログラム、記憶媒体

【請求項の数】 21

【発明者】

【住所又は居所】 東京都大田区下丸子3丁目30番2号 キヤノン株式会社
社内

【氏名】 三原 誠

【特許出願人】

【識別番号】 000001007

【氏名又は名称】 キヤノン株式会社

【代理人】

【識別番号】 100076428

【弁理士】

【氏名又は名称】 大塚 康德

【電話番号】 03-5276-3241

【選任した代理人】

【識別番号】 100112508

【弁理士】

【氏名又は名称】 高柳 司郎

【電話番号】 03-5276-3241

【選任した代理人】

【識別番号】 100115071

【弁理士】

【氏名又は名称】 大塚 康弘

【電話番号】 03-5276-3241

【選任した代理人】

【識別番号】 100116894

【弁理士】

【氏名又は名称】 木村 秀二

【電話番号】 03-5276-3241

【手数料の表示】

【予納台帳番号】 003458

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 0102485

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 ログ取得方法およびプログラム、記憶媒体

【特許請求の範囲】

【請求項 1】 所定の処理を行う関数を備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行う関数のアドレスを、ログ取得のための関数のアドレスに書き換える工程と、

前記所定の処理を行う関数を選択する工程と、を備え、

前記ログ取得のための関数は、

前記所定の処理を行う関数を呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記選択する工程において選択された関数を呼び出す際の所定の情報をログとして記録する工程と、

前記選択する工程において選択された関数の実行結果を受け取った際の所定の情報をログとして記録する工程と

を備えることを特徴とするログ取得方法。

【請求項 2】 所定の処理を行う関数を備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行う関数のアドレスを、ログ取得のための関数のアドレスに書き換える工程と、

前記所定の処理を行う関数を選択する工程と、を備え、

前記ログ取得のための関数は、

前記所定の処理を行う関数を呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記所定の処理を行う関数を呼び出す際の所定の情報と、該関数の実行結果を受け取った際の所定の情報とをログとして記録する工程と、

前記選択する工程において選択された関数について、ログの記録を停止する工程と

を備えることを特徴とするログ取得方法。

【請求項 3】 前記ログ取得のための関数は、前記選択する工程において選択された関数の実行時のエラーの有無を判断する工程を更に備え、

前記ログとして記録する工程は、前記エラーの有無を判断する工程によりエラー有りとは判断された場合、前記所定の情報をログとして記録することを特徴とする請求項 1 に記載のログ取得方法。

【請求項 4】 前記ログ取得のための関数は、前記選択する工程において選択された関数の実行時のエラーの有無を判断する工程を更に備え、

前記ログの記録を停止する工程は、前記エラーの有無を判断する工程によりエラー有りとは判断された場合、ログの記録を停止することを特徴とする請求項 2 に記載のログ取得方法。

【請求項 5】 所定の処理を行うメソッドを備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行うメソッドのアドレスを、ログ取得のためのメソッドのアドレスに書き換える工程と、

前記所定の処理を行うメソッドを選択する工程と、を備え、

前記ログ取得のためのメソッドは、

前記所定の処理を行うメソッドを呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記選択する工程において選択されたメソッドを呼び出す際の所定の情報をログとして記録する工程と、

前記選択する工程において選択されたメソッドの実行結果を受け取った際の所定の情報をログとして記録する工程と

を備えることを特徴とするログ取得方法。

【請求項 6】 所定の処理を行うメソッドを備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行うメソッドのアドレスを、ログ取得のためのメソッドのアドレスに書き換える工程と、

前記所定の処理を行うメソッドを選択する工程と、を備え、

前記ログ取得のためのメソッドは、

前記所定の処理を行うメソッドを呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記所定の処理を行うメソッドを呼び出す際の所定の情報と、該関数の実行結果を受け取った際の所定の情報とをログとして記録する工程と、

前記選択する工程において選択されたメソッドについて、ログの記録を停止する工程と

を備えることを特徴とするログ取得方法。

【請求項 7】 前記ログ取得のための関数は、前記選択する工程において選択されたメソッドの実行時のエラーの有無を判断する工程を更に備え、

前記ログとして記録する工程は、前記エラーの有無を判断する工程によりエラー有りとは判断された場合、前記所定の情報をログとして記録することを特徴とする請求項 5 に記載のログ取得方法。

【請求項 8】 前記ログ取得のための関数は、前記選択する工程において選択されたメソッドの実行時のエラーの有無を判断する工程を更に備え、

前記ログの記録を停止する工程は、前記エラーの有無を判断する工程によりエラー有りとは判断された場合、ログの記録を停止することを特徴とする請求項 6 に記載のログ取得方法。

【請求項 9】 COMサーバがエクスポートするインターフェースと、該インターフェースに属するメソッドとをツリー構造で表示する工程を更に備え、

前記選択する工程は、前記表示を介して選択されることを特徴とする請求項 5 または 6 に記載のログ取得方法。

【請求項 10】 前記選択する工程は、メソッド単位で選択可能であることを特徴とする請求項 9 に記載のログ取得方法。

【請求項 11】 前記選択する工程は、前記インターフェースを選択することで、該インターフェースに属する全てのメソッドを選択可能であることを特徴とする請求項 9 に記載のログ取得方法。

【請求項 12】 所定の処理を行う関数を備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行う関数のアドレスを、ログ取得のための関数

のアドレスに書き換える工程と、

前記所定の処理を行う関数を選択する工程と、を備え、

前記ログ取得のための関数は、

前記所定の処理を行う関数を呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記所定の処理を行う関数を呼び出す際の所定の情報と、前記実行結果を受け取った際の所定の情報とをログとして記録する工程と、を備え、

前記ログとして記録する工程は、日付毎にログを記録可能であることを特徴とするログ取得方法。

【請求項 13】 所定の処理を行う関数を備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行う関数のアドレスを、ログ取得のための関数のアドレスに書き換える工程と、

前記所定の処理を行う関数を選択する工程と、を備え、

前記ログ取得のための関数は、

前記所定の処理を行う関数を呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記所定の処理を行う関数を呼び出す際の所定の情報と、前記実行結果を受け取った際の所定の情報とをログとして記録する工程と、を備え、

前記ログとして記録する工程は、ログのサイズが一定サイズ以上になった場合に、新たに別ファイルを作成するとともに、作成日時の古いログを削除することを特徴とするログ取得方法。

【請求項 14】 所定の処理を行う関数を備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行う関数のアドレスを、ログ取得のための関数のアドレスに書き換える工程と、

前記所定の処理を行う関数を選択する工程と、を備え、

前記ログ取得のための関数は、

前記所定の処理を行う関数を呼び出し、該所定の処理を実行させ、受け取った

実行結果を前記プログラムに渡す工程と、

前記所定の処理を行う関数を呼び出す際の所定の情報と、前記実行結果を受け取った際の所定の情報とをログとして記録する工程と、を備え、

前記ログとして記録する工程は、ログの個数が一定数以上になった場合に、新たに別ファイルを作成するとともに、作成日時の古いログを削除することを特徴とするログ取得方法。

【請求項 15】 所定の処理を行う関数を備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行う関数のアドレスを、ログ取得のための関数のアドレスに書き換える工程と、

前記所定の処理を行う関数を選択する工程と、を備え、

前記ログ取得のための関数は、

前記所定の処理を行う関数を呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記所定の処理を行う関数を呼び出す際の所定の情報と、前記実行結果を受け取った際の所定の情報とをログとしてメモリに格納する工程と、を備え、

前記ログとしてメモリに格納する工程は、メモリ上のログの個数が一定数以上になった場合に、該メモリ上のログをディスク装置に移動させ、保存させることを特徴とするログ取得方法。

【請求項 16】 所定の処理を行うメソッドを備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行うメソッドのアドレスを、ログ取得のためのメソッドのアドレスに書き換える工程と、

前記所定の処理を行うメソッドを選択する工程と、を備え、

前記ログ取得のためのメソッドは、

前記所定の処理を行うメソッドを呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記所定の処理を行うメソッドを呼び出す際の所定の情報と、前記実行結果を受け取った際の所定の情報とをログとして記録する工程と、を備え、

前記ログとして記録する工程は、日付毎にログを記録可能であることを特徴とするログ取得方法。

【請求項 17】 所定の処理を行うメソッドを備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行うメソッドのアドレスを、ログ取得のためのメソッドのアドレスに書き換える工程と、

前記所定の処理を行うメソッドを選択する工程と、を備え、

前記ログ取得のためのメソッドは、

前記所定の処理を行うメソッドを呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記所定の処理を行うメソッドを呼び出す際の所定の情報と、前記実行結果を受け取った際の所定の情報とをログとして記録する工程と、を備え、

前記ログとして記録する工程は、ログのサイズが一定サイズ以上になった場合に、新たに別ファイルを作成するとともに、作成日時の古いログを削除することを特徴とするログ取得方法。

【請求項 18】 所定の処理を行うメソッドを備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行うメソッドのアドレスを、ログ取得のためのメソッドのアドレスに書き換える工程と、

前記所定の処理を行うメソッドを選択する工程と、を備え、

前記ログ取得のためのメソッドは、

前記所定の処理を行うメソッドを呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記所定の処理を行うメソッドを呼び出す際の所定の情報と、前記実行結果を受け取った際の所定の情報とをログとして記録する工程と、を備え、

前記ログとして記録する工程は、ログの個数が一定数以上になった場合に、新たに別ファイルを作成するとともに、作成日時の古いログを削除することを特徴とするログ取得方法。

【請求項 19】 所定の処理を行うメソッドを備えるプログラムの実行中の

ログを取得するログ取得方法であって、

ロードされた前記所定の処理を行うメソッドのアドレスを、ログ取得のためのメソッドのアドレスに書き換える工程と、

前記所定の処理を行うメソッドを選択する工程と、を備え、

前記ログ取得のためのメソッドは、

前記所定の処理を行うメソッドを呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記所定の処理を行うメソッドを呼び出す際の所定の情報と、前記実行結果を受け取った際の所定の情報とをログとしてメモリに格納する工程と、を備え、

前記ログとしてメモリに格納する工程は、メモリ上のログの個数が一定数以上になった場合に、該メモリ上のログをディスク装置に移動させ、保存させることを特徴とするログ取得方法。

【請求項 20】 請求項 1 乃至 19 のいずれか 1 つに記載のログ取得方法をコンピュータによって実現させるための制御プログラムを格納した記憶媒体。

【請求項 21】 請求項 1 乃至 19 のいずれか 1 つに記載のログ取得方法をコンピュータによって実現させるための制御プログラム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、複数にモジュール分けされたソフトウェアの処理ログを取得するための技術に関するものである。

【0002】

【従来の技術】

従来より、再現率の低いソフトウェアの障害に対しては、ソフトウェアの処理ログを取得し、当該処理ログを解析することによって、障害の原因をつきとめ、その対策を講じてきた。

【0003】

【発明が解決しようとする課題】

しかし、上記従来の処理ログの取得には以下のような問題点がある。

(1) ユーザの動作環境でもログを取得しつづけるためには、ソフトウェアのモジュール自体に手を加え、処理ログ取得ルーチンを追加しなければならず、処理ログ取得のための作業負荷が大きかった。

(2) 処理ログ取得はモジュール毎に行うため、生成されたログはモジュール単位のものとなってしまい、ソフトウェア全体の処理を、完全に時間順のログとして取得するのが困難である。このため、全体の処理ログとしての見通しが悪く、ログを解析して障害の原因を発見するまでのプロセスに工数がかかっていた。

【0004】

本発明は、上記課題を鑑みてなされたものであり、複数のモジュール分けされたソフトウェアの処理ログを容易に取得でき、かつ、ソフトウェアの障害の原因の解析のための工数を削減することが可能なログ取得方法、ならびに該方法をコンピュータによって実現させるためのプログラム、および該プログラムを格納した記憶媒体を提供することを目的とする。

【0005】

【課題を解決するための手段】

上記の目的を達成するために本発明に係るログ取得方法は以下のような構成を備える。即ち、

所定の処理を行う関数を備えるプログラムの実行中のログを取得するログ取得方法であって、

ロードされた前記所定の処理を行う関数のアドレスを、ログ取得のための関数のアドレスに書き換える工程と、

前記所定の処理を行う関数を選択する工程と、を備え、

前記ログ取得のための関数は、

前記所定の処理を行う関数を呼び出し、該所定の処理を実行させ、受け取った実行結果を前記プログラムに渡す工程と、

前記選択する工程において選択された関数を呼び出す際の所定の情報をログとして記録する工程と、

前記選択する工程において選択された関数の実行結果を受け取った際の所定の情報をログとして記録する工程とを備える。

【0006】

【発明の実施の形態】

【第1の実施形態】

本実施形態は、あるモジュールから別のモジュール内に存在する関数の呼び出しが行われる際の仕組みである、メモリに保持されたインポート関数アドレス、または仮想関数アドレステーブル（Virtual Address Table）を利用して、モジュール間の関数呼び出しをフックしてログに記録することで、ソフトウェアのモジュール自体に手を加えることなく、ソフトウェア全体の処理を、時間順のログとして取得することを可能にするものである。以下に具体的に説明する。

【0007】

＜システム構成＞

図1は、本発明の各実施形態にかかるログ取得方法を実現するコンピュータ（ソフトウェア評価システム）の構成をあらわす図である。説明を簡略化するために、本実施形態では、本ソフトウェア評価システムが1台のPC内部に構築されるものとするが、本発明のログ取得方法の特徴は1台のPC内部に構築されるか、あるいは複数のPCにネットワークシステムとして構築されるかによらず有効である。

【0008】

本ソフトウェア評価システムを搭載するコンピュータには、CPU1、チップセット2、RAM3、ハードディスクコントローラ4、ディスプレイコントローラ5、ハードディスクドライブ6、CD-ROMドライブ7、ディスプレイ8が搭載されている。また、CPUとチップセットを繋ぐ信号線11、チップセット2とRAM3とを繋ぐ信号線12、チップセット2と各種周辺機器とを繋ぐ周辺機器バス13、ハードディスクコントローラ4とハードディスクドライブ6とを繋ぐ信号線14、ハードディスクコントローラ4とCD-ROMドライブ7とを繋ぐ信号線15、ディスプレイコントローラ5とディスプレイ8とを繋ぐ信号線16が搭載されている。

【0009】

＜関数処理に対する処理ログ取得＞

本発明の第 1 の実施形態にかかるログ取得方法を実現するソフトウェア評価システムを説明するために、まず図 2 によって、複数のモジュールに分かれたソフトウェアが、通常の状態でどのようにメモリにロードされるかを説明する。

【0010】

通常、複数のモジュールに分かれたソフトウェアは、全体の制御を行う実行ファイル EXE (23) と、モジュールとして存在し EXE の補完的な役割を担うダイナミックリンクライブラリ DLL (27) とに分かれており、メモリには EXE と DLL の両方がロードされる。EXE はコードセグメント (28) とデータセグメント (29)、そしてインポート関数アドレステーブル (22) から成っている。更に、インポート関数アドレステーブルは、関数の所属する DLL によって分かれており (21, 24)、DLL ごとにそれぞれの関数がロードされたアドレスが書かれている (30～35)。DLL の関数の実体は、DLL ごとに分かれて (25, 26) ロードされ、それぞれの関数は該当する DLL の一部としてロードされる (36～41)。この図では、1 本の EXE が A.DLL 及び B.DLL の 2 つのダイナミックリンクライブラリ内の関数を使用している例を示しており、実際に使用される関数は Func AA, Func AB, Func AC, Func BA, Func BB, Func BC の 6 個となっている。

【0011】

EXE のコードセグメント内にあるコードが関数 Func AA を呼び出す場合には、まずインポート関数アドレステーブル内に書かれた Func AA のアドレス (30) が読み込まれる。ここには実際には A.DLL の一部として読み込まれた Func AA コード (36) のアドレスが書かれており、そのアドレスをコールすることによって、EXE のコードは A.DLL の Func AA を呼び出すことができる。

【0012】

図 3 は、本発明の第 1 の実施形態にかかるログ取得方法を実現するソフトウェア評価システムのメモリ構成をあらわす図であり、図 2 とは、ログ取得用のコードに対して IAT Patch (Import Address Table

Patch)という手法を用いて、関数呼び出しをリダイレクトしているという点で異なっている。

【0013】

ログ取得が開始されると、メモリ内にはIAT Patch用のDLLであるC.DLL(58)がロードされる。C.DLLはインポート関数アドレステーブル(52)内に書かれた関数のアドレスを、C.DLL内のログ取得コードであるFunc CAA, Func CAB, Func CAC, Func CBA, Func CBB, Func CBCのアドレスに書き換える(61~66)。C.DLL内のFunc CAA, Func CAB, Func CAC, Func CBA, Func CBB, Func CBCのコード(73~78)は、ログを記録すると共に、元々の関数呼び出しを受けるべくメモリにロードされている、該当する関数であるFunc AA, Func AB, Func AC, Func BA, Func BB, Func BC(67~72)を呼び出す。

【0014】

図4Aは、図3におけるIAT Patchの処理をあらわす図、図4Bはログ取得処理の流れを示すフローチャートである。説明を簡略化するために、この図ではEXEがA.DLL内のFunc AAを呼び出す際に、IAT Patchによるログ取得コードがどのように動作するかの例をあらわしている。

【0015】

EXE(図4Aの91)がFunc AAをコールすると(図4Aの94)、C.DLL内にあるログ取得コードがDLL名/関数名をメモリに保存し(図4BのステップS402)、呼び出し時刻をメモリに保存し、呼び出し時のパラメータをメモリに保存し、呼び出し時のポインタパラメータの指すメモリ内容を、別メモリに保存する(図4Aの95、図4BのステップS403)。その後C.DLLは本来呼び出されるはずであった、A.DLL(図4Aの93)内のFunc AAをコールする(図4Aの96、図4BのステップS404)。A.DLLのFunc AA処理(図4Aの97)が終了し、C.DLLに制御がリターンすると(図4Aの98)、C.DLLはリターン時の時刻をメモリに保存し、戻り値をメモリ

に保存し、リターン時にポインタパラメータが指すメモリ内容を、別メモリに保存する(図4Aの99)。その後、C.DLLは保存したログ情報をファイルに書き込み(図4Aの100、図4BのステップS405)、あたかもA.DLLのFunc AAが通常通りに終了したかのように、EXEにリターンする(101)。

【0016】

図5は、本発明の第1の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの内部構成をあらわす図である。通常は実行形式のEXE(113)が、DLL-1(116)やDLL-2(117)内の関数を呼び出すが、ここではAPIトレーサと呼ばれるログ取得コードを埋め込み(114)、処理ログを生成している(115)。APIトレーサは、DLL-1やDLL-2の関数定義を記述したファイル(111)と、どのDLLのどの関数のインポート関数テーブルを書き換えてログを取得するかの設定シナリオ(トレースシナリオ112)を元に動作する。

【0017】

<メソッド処理に対する処理ログ取得>

次に、本発明の第1の実施形態にかかるログ取得方法を実現するソフトウェア評価システムにおいて、実行ファイルEXE(118)がCOM(Component Object Model)サーバでエクスポートされているインターフェースのインスタンス作成時に、どのようにメモリにロードされるかを説明するために、まず、図6によって、通常の状態でどのようにメモリにロードされるかを説明する。

【0018】

通常、インターフェースのインスタンス作成を行うと、COMサーバ内で、要求されたインターフェース(121, 122)と、そのメソッド(:オブジェクト指向プログラミングにおいて、オブジェクトの実行する手続きを記述したプログラム、130~135)が作成され、それらは、メモリ上に両方がロードされる。ここで、virtual address table(仮想アドレステーブル)は作成された各インターフェース毎に作られ(118, 120)、作成要求

を行ったEXEに渡される。このvirtual address tableには各メソッドの作成されたアドレスが書かれている(124~129)。EXEはこれら情報を利用し、各インターフェースに対して呼び出しを行う。この図では、1本のEXEがInterface A及びInterface Bの2つのインターフェースのインスタンスを作成しており、そのインターフェース内部のメソッドを使用している例を示しており、実際に使用されているメソッドは、Method AA, Method AB, Method AC, Method BA, Method BB, Method BCとなっている。

【0019】

EXEのコードが関数Method AAを呼び出す場合には、まずvirtual address table内に書かれたMethod AAのアドレス(124)が読み込まれる。ここには実際にはCOMサーバのInterface Aの一部として作成されたMethod AAコード(130)のアドレスが書かれており、そのアドレスをコールすることによって、EXEのコードはInterface AのMethod AAを呼び出すことができる。

【0020】

図7は、本発明の第1の実施形態にかかるログ取得方法を実現するソフトウェア評価システムのメモリ構成をあらわす図であり、図6とは、ログ取得用のコードに対してVTable Patch(virtual address table Patch)という手法を用いて、メソッド呼び出しをリダイレクトしているという点で異なっている。

【0021】

ログ取得が開始されると、メモリ内にはVTable Patch用のDLL(143)がロードされる。このDLLはvirtual address table(136, 138)内に書かれたメソッドのアドレスを、DLL内のログ取得コードであるMethod A'A, Method A'B, Method A'C, Method B'A, Method B'B, Method B'Cのアドレスに書き換える(145~150)。DLL内のMethod A'A, Method A'B, Method A'C, Method B'A, Me

Method B'B, Method B'Cのコード(157~162)は、ログを記録すると共に、元々のメソッド呼び出しを受けるべくメモリにロードされている、該当するメソッドであるMethod AA, Method AB, Method AC, Method BA, Method BB, Method BC(157~162)を呼び出す。

【0022】

図8Aは、図7におけるVTable Patchの処理をあらわす図、図8Bはログ取得処理の流れを示すフローチャートである。説明を簡略化するために、この図ではEXEがCOMサーバ内のInterface AのMethod AAを呼び出す際に、VTable Patchによるログ取得コードがどのように動作するか例をあらわしている。

【0023】

EXE(図8Aの163)がMethod AAをコールすると(図8Aの166)、DLL内にあるログ取得コードがモジュール名/インターフェース名/メソッド名をメモリに保存し(図8BのステップS802)、呼び出し時刻をメモリに保存し、呼び出し時のパラメータをメモリに保存し、呼び出し時のポインタパラメータの指すメモリ内容を、別メモリに保存する(図8Aの167、図8BのステップS803)。その後DLLは本来呼び出されるはずであった、COMサーバ(図8Aの165)内のMethod AAをコールする(図8Aの168、図8BのステップS804)。COMサーバのMethod AA処理(図8Aの169)が終了し、DLLに制御がリターンすると(図8Aの170)、DLLはリターン時の時刻をメモリに保存し、戻り値をメモリに保存し、リターン時にポインタパラメータが指すメモリ内容を、別メモリに保存する(図8Aの171)。その後、DLLは保存したログ情報をファイルに書き込み(図8Aの172、図8BのステップS805)、あたかもCOMサーバのMethod AAが通常通りに終了したかのように、EXEにリターンする(図8Aの173)。

【0024】

図9は、本発明の第1の実施形態にかかるソフトウェア評価システムの内部構成をあらわす図である。通常は実行形式のEXE(176)が、COMサーバ1(

179)やCOMサーバ2(180)内のメソッドを呼び出すが、ここではAPIトレーサと呼ばれるログ取得コードを埋め込み(177)、処理ログを生成している(178)。APIトレーサは、COMサーバ1(179)やCOMサーバ2の関数定義を記述したファイル(174)と、どのCOMサーバのどのインターフェースのどのメソッドのvirtual address tableを書き換えてログを取得するかの設定シナリオ(175)を元に動作する。

【0025】

<ユーザインターフェース>

図10は、本発明の第1の実施形態にかかるログ取得方法を実現するソフトウェア評価システムで、ログ取得を開始するための関数/メソッドを設定するためのユーザインターフェースである。

【0026】

ユーザインターフェースには、ログ取得対象となっているモジュール/インターフェース(181)から、どのモジュール/インターフェースを設定するかを選択するドロップダウンリスト(183, 184)と、そこで選択したモジュール/インターフェースでエクスポートされている関数/メソッド(182)から、どの関数/メソッドを設定するかを選択するドロップダウンリスト(185, 186)とを備える。

【0027】

図11は、本発明の第1の実施形態にかかるログ取得方法を実現するソフトウェア評価システムで、図10で設定した内容に従い、ログを取得する場合の流れを示す図である。

【0028】

処理が開始されると(ステップS1)、ログ取得コードは呼び出された関数/メソッドがログ取得開始トリガーとして設定されているか判別する(ステップS2)。ログ取得開始トリガーと一致した場合には、ログ取得を開始し、モジュール名、インターフェース名、関数/メソッド名をHDDに保存する(ステップS3)。次にログ取得コードは、呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をHDDに保存し(ステップS4)、元の関数/メソッドを

呼び出す(ステップ S 5)。関数／メソッドからリターンすると、ログ取得コードはリターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容を HDD に保存する(ステップ S 6)。この処理はユーザから終了の命令があるまで(ステップ S 7)続行されるが、その際には、ログ取得開始トリガーの判別を行わない。

【0029】

以上の説明から明らかなように、本実施形態によれば、複数のモジュール分けされたソフトウェアの処理ログの取得において、モジュール自体に変更を加えることなく、モジュール内に用意された任意の関数／メソッドから処理ログを容易に取得することが可能となり、ソフトウェアにおける障害の原因の解析のための工数を削減する効果を与える。さらにログ取得を行う関数／メソッドを任意に選択できるようにすることで、ログの解析が容易になるという効果がもたらされる。

【0030】

【第2の実施形態】

上記第1の実施形態では、ログを取得する関数／メソッドをユーザが任意に選択することが可能なユーザインターフェースについて説明したが、本実施形態では、ログ取得を停止する関数／メソッドを任意に選択可能なユーザインターフェースについて説明する。

【0031】

図12は、本実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、ログ取得を停止するための関数／メソッドを設定するためのユーザインターフェースである。

【0032】

ユーザインターフェースには、ログ取得対象となっているモジュール／インターフェース(187)から、どのモジュール／インターフェースを設定するかを選択するドロップダウンリスト(189, 190)と、そこで選択したモジュール／インターフェースでエクスポートされている関数／メソッド(188)から、どの関数／メソッドを設定するかを選択するドロップダウンリスト(191, 192)

とを備える。

【0033】

図13は、本実施形態にかかるログ取得方法を実現するソフトウェア評価システムで、図12で設定した内容に従い、ログを取得する場合の流れを示す図である。

【0034】

処理が開始されると(ステップS9)、ログ取得を開始し、モジュール名、インターフェース名、関数／メソッド名をHDDに保存する(ステップS10)。次にログ取得コードは、呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をHDDに保存し(ステップS11)、元の関数／メソッドを呼び出す(ステップS12)。関数／メソッドからリターンすると、ログ取得コードはリターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容をHDDに保存する(ステップS13)。その後呼び出された関数／メソッドがログ取得停止トリガーとして設定されているか判別する(ステップS14)。ログ取得停止トリガーと一致した場合には、ログ取得処理は終了する(ステップS16)。また、ログ取得停止トリガーと一致しない場合でも、この処理はユーザから終了の命令があると(ステップS15)終了される。

【0035】

これにより、任意の関数／メソッドのログ取得を停止することができることとなり、ユーザの求めるログのみを取得できるため、ログの解析が容易になるという効果がもたらされる。

【0036】

【第3の実施形態】

上記第1および第2の実施形態において述べたユーザインタフェースでは、ユーザが選択した任意の関数／メソッドでログの取得開始／停止をユーザが任意に選択することとしたが、ユーザが選択した任意の関数／メソッドが、エラーで終了した場合のみ、ログの取得開始／停止をするように設定することも可能である。

【0037】

図14は、本実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、ログ取得を開始／停止するための関数／メソッドを設定するためのユーザインターフェースに、エラーで終了した場合のみトリガー機能（ログ取得開始／停止機能）を使用する設定を追加したものである。本ユーザインターフェースは、図10及び、図12双方に適應することが可能である。

【0038】

ユーザインターフェースには、ログ取得対象となっているモジュール／インターフェース(193)から、どのモジュール／インターフェースを設定するかを選択するドロップダウンリスト(195, 196)と、そこで選択したモジュール／インターフェースでエクスポートされている関数／メソッド(194)から、どの関数／メソッドを設定するかを選択するドロップダウンリスト(197, 198)と、関数／メソッドがエラーで終了した場合のみトリガー機能を有効にする為のチェックボックスとを備える。

【0039】

図15は、本実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、関数／メソッドのエラー定義の内容を示したものである。本実施形態においてはエラー定義がファイルとなっており、ファイルには各関数／メソッドのパラメータ及び戻り値とそのエラー条件等が定義され記されている。

【0040】

図16は、本実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、図14で設定した内容に従い、ログを取得する場合の流れを示す図であり、ログ取得開始と、エラー時のみトリガーを使用する機能を併用して設定した場合である。

【0041】

処理が開始されると(ステップS17)、ログ取得コードは呼び出された関数／メソッドがログ取得開始トリガーとして設定されているか判別する(ステップS18)。ログ取得開始トリガーとして設定されている場合には、モジュール名、インターフェース名、関数／メソッド名をメモリに一時的に保存する(ステップS19)。

【0042】

次にログ取得コードは、呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をメモリに一時的に保存し(ステップS20)、元の関数／メソッドを呼び出す(ステップS21)。メソッドからリターンすると、ログ取得コードはリターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容をメモリに一時的に保存する(ステップS22)。

【0043】

次にログ取得開始トリガーがエラーでのみ使用するかどうかを判別し(ステップS23)、そうであれば、関数／メソッドがエラーであるかを判別する(ステップS24)。もし、エラーでない場合には、一時メモリに保存したログを破棄し(ステップS25)、処理のはじめに戻る。関数／メソッドがエラーであった場合には、一時メモリに保存したログをHDDに保存し(ステップS26)、通常のログ取得処理を引き続き行う(ステップS27)。この処理はユーザから終了の命令があるまで(ステップS28)続行される。

【0044】

図17は、本実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、図16のステップS27で示した通常のログ取得処理の詳細の流れを示す図である。

【0045】

処理が開始されると(ステップS30)、ログ取得を開始し、モジュール名、インターフェース名、関数／メソッド名をHDDに保存する(ステップS31)。次にログ取得コードは、呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をHDDに保存し(ステップS32)、元の関数／メソッドを呼び出す(ステップS33)。関数／メソッドからリターンすると、ログ取得コードはリターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容をHDDに保存する(ステップS34)。

【0046】

図18は、本実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、図14で設定した内容に従い、ログを取得する場合の流れを示す図であ

り、ログ取得停止と、エラー時のみトリガーを使用する機能とを併用して設定した場合である。

【0047】

処理が開始されると(ステップS40)、ログ取得を開始し、モジュール名、インターフェース名、関数／メソッド名をHDDに保存する(ステップS41)。次にログ取得コードは、呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をHDDに保存し(ステップS42)、元の関数／メソッドを呼び出す(ステップS43)。関数／メソッドからリターンすると、ログ取得コードはリターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容をHDDに保存する(ステップS34)。その後呼び出された関数／メソッドがログ取得停止トリガーとして設定されているか判別する(ステップS45)。ログ取得停止トリガーと一致した場合には、次にログ取得停止トリガーがエラーでのみ使用するかどうかを判別し(ステップS46)、そうであれば、関数／メソッドがエラーであるかを判別する(ステップS47)。もし、エラーである場合は、ログ取得処理は終了する(ステップS48)。また、この処理はユーザから終了の命令があると(ステップS49)終了される。

【0048】

これにより、任意の関数／メソッドでエラーが発生した場合からログの取得を開始したり、停止したりすることが可能となり、ユーザの求めるログを取得することが可能になり、ログの解析が容易になるという効果がもたらされる。

【0049】

【第4の実施形態】

上記第1乃至第3の実施形態におけるユーザインターフェースは、選択可能な関数／メソッドが所定の順序で配列されているにすぎなかったが、インターフェース、メソッドの関係が容易に把握できるようなツリー表示としてもよい。

【0050】

図19は、本発明の第4の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、インターフェースとメソッドのツリー表示を行うユーザインターフェースに関するものである。

【0051】

ユーザインターフェースには、インターフェース及びメソッドをツリー表示する為のビュー(200)が備えられている。ユーザがインターフェースInterfaceA(201)をチェックすることで、そのインターフェース内の全てのメソッドMethodAA, MethodAB, MethodAC(202~204)は全て選択され、ログの取得対象となる。また、インターフェースInterfaceB(205)のチェックを外すことで、そのインターフェース内の全てのメソッドMethodBA, MethodBB, MethodBC(205~208)は全て選択解除され、ログの取得対象外となる。

【0052】

図20は、本実施形態にかかるソフトウェア評価システムの、図19のようにログ取得対象を選択した場合の、ログを取得する際の処理の流れを示す図である。

【0053】

処理が開始されると(ステップS51)、ログ取得コードはインターフェースの何らかのメソッド呼び出しが行われるたびに、そのメソッドのインターフェースが、ログ取得対象となっているかを判別する(ステップS52)。ログ取得対象となっていた場合には、モジュール名とインターフェース、メソッド名をHDDに保存する(ステップS53)。

【0054】

次にログ取得コードは、呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をHDDに保存し(ステップS54)、元のメソッドを呼び出す(ステップS55)。メソッドからリターンすると、ログ取得コードはリターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容をHDDに保存する(ステップS56)。この処理はユーザから終了の命令があるまで(ステップS57)続行される。

【0055】

これにより、インターフェース、メソッドの関係を更に簡便に知ることができ、また、ログを取得したいインターフェースを容易に選択可能となり、ユーザの

、求めるログを取得する方法が簡単になるという効果がもたらされる。

【0056】

【第5の実施形態】

上記第1乃至第3の実施形態では、任意のメソッドを選択するにあたり、インターフェース単位で選択を行うこととしたが、さらに、メソッド単位で選択を行うことも可能である。

【0057】

図21は、本実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、インターフェースとメソッドのツリー表示を行うユーザインターフェースに関するものであり、図19と同様であるが、選択方法が異なっている。

【0058】

ユーザインターフェースには、インターフェース及びメソッドをツリー表示する為のビュー(209)が備えられている。ユーザがメソッドMethodAA(211)、MethodAC(213)、MethodBA(215)をチェックし、MethodAB(212)、MethodBB(216)、MethodBC(217)のチェックを外すことで、インターフェースInterfaceA(210)、InterfaceB(214)内の全てのメソッドではなく、各インターフェースのチェックしたメソッドのみをログの取得対象とすることが可能となる。

【0059】

図22は、本実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、図21のようにログ取得対象を選択した場合の、ログを取得する際の処理の流れを示す図である。

【0060】

処理が開始されると(ステップS59)、ログ取得コードはインターフェースの何らかのメソッド呼び出しが行われるたびに、そのインターフェースのメソッドが、ログ取得対象となっているかを判別する(ステップS60)。ログ取得対象となっていた場合には、モジュール名とインターフェース、メソッド名をHDDに保存する(ステップS61)。

【 0 0 6 1 】

次にログ取得コードは、呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をHDDに保存し(ステップS 6 2)、元のメソッドを呼び出す(ステップS 6 3)。メソッドからリターンすると、ログ取得コードはリターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容をHDDに保存する(ステップS 6 4)。この処理はユーザから終了の命令があるまで(ステップS 6 5)続行される。

【 0 0 6 2 】

これにより、インターフェース毎という大きな範囲ではなく、ログを取得したいメソッドをメソッド単位でそれぞれ容易に選択可能となり、ユーザの求めるログを簡単に取得することができるという効果がもたらされる。

【 0 0 6 3 】**【第 6 の実施形態】**

上記各実施形態においては、取得したログをHDDの任意の場所に保存していたが、ログの解析を容易に行うことができるよう、日付ごとに保存するようにしてもよい。

【 0 0 6 4 】

図 2 3 は、日付毎にログを分割保存する場合の処理の流れを示す図である。

【 0 0 6 5 】

処理が開始されると(ステップS 7 1)、ログ取得を開始し、モジュール名、インターフェース名、関数／メソッド名をメモリに保存する(ステップS 7 2)。

【 0 0 6 6 】

次にログ取得コードは、呼び出し時の日付・時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をメモリに保存し(ステップS 7 3)、元の関数／メソッドを呼び出す(ステップS 7 4)。関数／メソッドからリターンすると、ログ取得コードはリターン時の日付・時刻、戻り値及びポインタ・パラメータの指すメモリの内容をメモリに保存する(ステップS 7 5)。その後呼び出された関数／メソッドのリターン時の日付が、前回に保存したログのリターン時の日付と異なるかを判別する(ステップS 7 6)。日付が異なる場合には、新規のログファイ

ルを作成し、ログをそのファイルに保存する(ステップS77)、同じ場合には、従来のログファイルにログを保存する(ステップS78)。この処理はユーザから終了の命令があると(ステップS79)終了される(ステップS80)。

【0067】

これにより、ユーザは求めるログを日付毎に取得することができ、ログの解析が容易になるという効果がもたらされる。

【0068】

【第7の実施形態】

上記第6の実施形態では、日付ごとにログを保存する場合について述べたが、サイズ毎もしくは個数毎にログを分割して保存するようにしてもよい。

【0069】

図24は、サイズ毎もしくは個数毎にログを分割保存する場合の処理の流れを示す図である。

【0070】

処理が開始されると(ステップS71)、ログ取得を開始し、モジュール名、インターフェース名、関数/メソッド名をメモリに保存する(ステップS72)。次にログ取得コードは、呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をメモリに保存し(ステップS73)、元の関数/メソッドを呼び出す(ステップS74)。関数/メソッドからリターンすると、ログ取得コードはリターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容をメモリに保存する(ステップS75)。その後新規のログデータを従来のログファイルに保存した場合に、ファイルのサイズが一定値を超えるかもしくは一定の個数のログを超えるかどうかを判別する(ステップS76)。

【0071】

ある一定サイズを超えてしまう場合もしくはある一定の個数を超えてしまう場合には、新規のログファイルを作成し、ログをそのファイルに保存する(ステップS77)、同じ場合には、従来のログファイルにログを保存する(ステップS78)。新規のログファイルを作成した場合には、リングバッファを使用する場合で且つ、作成したログファイルの数が2個より多いかどうかを判別する(ステッ

プS79)。もし、条件に当てはまる場合には、最も古いログファイルを削除する(ステップS90)。この処理はユーザから終了の命令があると(ステップS91)終了される(ステップS92)。

【0072】

これにより、ユーザは複数作成される各ログを一定サイズ以内で取得することや、一定のログの個数毎で取得ができ、取り扱いが容易になる効果がもたらされる。また、リングバッファを使用することで、本ソフトウェア評価システムがPCのリソースに対して与える負荷を一定に押さえる事が可能になり、安定したログの取得を行えるようになる。

【0073】

【第8の実施形態】

図25は、取得したログを一定数だけメモリに保存する場合のメモリの概要図である。

【0074】

一定数のログを保存するためのログ格納メモリ領域がn個存在し(218)、各ログ格納メモリ領域には、1回分の関数/メソッドのログが格納され、その情報はモジュール名、インターフェース名、関数/メソッド名、呼び出し時の時刻、呼び出し時のパラメータデータ、終了時の時刻、終了時のパラメータデータ、戻り値データ等が格納されており(219)、可変サイズとなる。このメモリ領域には、ログ格納メモリ領域1から順にログデータは格納されていき、ログ格納メモリ領域nまで、使用し終わると、再度、ログ格納メモリ領域1から上書きしてログを格納する。

【0075】

図26は、取得したログを一定数だけメモリに保存する場合のログ取得の流れを示す図である。

【0076】

処理が開始されると(ステップS93)、ログ格納メモリ領域の場所を示す変数xを1に初期化する(ステップS94)。そして、ログ取得を開始し、モジュール名、インターフェース名、関数/メソッド名をログ格納メモリ領域xに保存する

(ステップS95)。

【0077】

次にログ取得コードは、呼び出し時の時刻、パラメータ及びポインタ・パラメータの指すメモリの内容をログ格納メモリ領域xに保存し(ステップS96)、元の関数／メソッドを呼び出す(ステップS97)。関数／メソッドからリターンすると、ログ取得コードはリターン時の時刻、戻り値及びポインタ・パラメータの指すメモリの内容をログ格納メモリ領域xに保存する(ステップS98)。その後ログ格納メモリ領域の場所を示す変数xに1を加算し(ステップS99)、xがログ格納メモリ領域数のnより大きいかを判別する(ステップS100)。

【0078】

xがnより大きい場合にはxに1を代入し、ログ格納メモリ領域の先頭から再度使用するように設定する(ステップS101)。その後、リングバッファを使用するかを判別し(ステップS102)、使用しない場合には、メモリ上に有る、全てのログデータをログファイルに保存し(ステップS103)、メモリ上のログデータを全て削除する(ステップS104)。この処理はユーザから終了の命令があると(ステップS105)終了される(ステップS106)。

【0079】

これにより、メモリの使用量をある程度の値でとどめることが可能となり、本ソフトウェア評価システムがPCのリソースに対して与える負荷を抑えることが可能になり、安定したログの取得を行えるようになる。

【0080】

【他の実施形態】

なお、本発明は、複数の機器(例えばホストコンピュータ、インタフェイス機器、リーダ、プリンタなど)から構成されるシステムに適用しても、一つの機器からなる装置(例えば、複写機、ファクシミリ装置など)に適用してもよい。

【0081】

また、本発明の目的は、前述した実施形態の機能を実現するソフトウェアのプログラムコードを記録した記憶媒体を、システムあるいは装置に供給し、そのシステムあるいは装置のコンピュータ(またはCPUやMPU)が記憶媒体に格納

されたプログラムコードを読み出し実行することによっても、達成されることは言うまでもない。

【0082】

この場合、記憶媒体から読み出されたプログラムコード自体が前述した実施形態の機能を実現することになり、そのプログラムコードを記憶した記憶媒体は本発明を構成することになる。

【0083】

プログラムコードを供給するための記憶媒体としては、例えば、フロッピー（登録商標）ディスク、ハードディスク、光ディスク、光磁気ディスク、CD-ROM、CD-R、磁気テープ、不揮発性のメモリカード、ROMなどを用いることができる。

【0084】

また、コンピュータが読み出したプログラムコードを実行することにより、前述した実施形態の機能が実現されるだけでなく、そのプログラムコードの指示に基づき、コンピュータ上で稼働しているOS（オペレーティングシステム）などが実際の処理の一部または全部を行い、その処理によって前述した実施形態の機能が実現される場合も含まれることは言うまでもない。

【0085】

さらに、記憶媒体から読み出されたプログラムコードが、コンピュータに挿入された機能拡張ボードやコンピュータに接続された機能拡張ユニットに備わるメモリに書込まれた後、そのプログラムコードの指示に基づき、その機能拡張ボードや機能拡張ユニットに備わるCPUなどが実際の処理の一部または全部を行い、その処理によって前述した実施形態の機能が実現される場合も含まれることは言うまでもない。

【0086】

【発明の効果】

以上説明したように本発明によれば、複数にモジュール分けされたソフトウェアの処理ログを容易に取得でき、かつ、ソフトウェアの障害の原因の解析のための工数を削減することが可能となる。

【図面の簡単な説明】**【図 1】**

本発明の第 1 の実施形態にかかるログ取得方法を実現するコンピュータ（ソフトウェア評価システム）の構成をあらわす図である。

【図 2】

本発明の第 1 の実施形態にかかるログ取得方法を説明するためにあらわした、関数ロード時の通常のメモリ構成をあらわす図である。

【図 3】

本発明の第 1 の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの I A T P a t c h 使用時のメモリ構成をあらわす図である。

【図 4 A】

本発明の第 1 の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの I A T P a t c h を使用時の状態を示す図である。

【図 4 B】

本発明の第 1 の実施形態にかかるログ取得方法を実現するソフトウェア評価システムのログ取得処理の流れを示す図である。

【図 5】

本発明の第 1 の実施形態にかかるソフトウェア評価システムの I A T P a t c h を使用時の内部構成をあらわす図である。

【図 6】

本発明の第 1 の実施形態にかかるログ取得方法を説明するためにあらわした、COMサーバのインターフェースのインスタンス作成時の通常のメモリ構成をあらわす図である。

【図 7】

本発明の第 1 の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの V T a b l e P a t c h 使用時のメモリ構成をあらわす図である。

【図 8 A】

本発明の第 1 の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの V T a b l e P a t c h を使用時の状態を示す図である。

【図 8 B】

本発明の第 1 の実施形態にかかるログ取得方法を実現するソフトウェア評価システムのログ取得処理の流れを示す図である。

【図 9】

本発明の第 1 の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、内部構成をあらわす図である。

【図 10】

本発明の第 1 の実施形態にかかるログ取得方法を実現するソフトウェア評価システムで、ログ取得を開始するための関数／メソッドを設定するためのユーザインターフェースを示す図である。

【図 11】

本発明の第 1 の実施形態にかかるログ取得方法を実現するソフトウェア評価システムで、図 10 で設定した内容に従い、ログを取得する場合の流れを示す図である。

【図 12】

本発明の第 2 に実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、ログ取得を停止するための関数／メソッドを設定するためのユーザインターフェースを示す図である。

【図 13】

本発明の第 2 の実施形態にかかるログ取得方法を実現するソフトウェア評価システムで、図 12 で設定した内容に従い、ログを取得する場合の流れを示す図である。

【図 14】

本発明の第 3 の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、ログ取得を開始／停止するための関数／メソッドを設定するためのユーザインターフェースに、エラーで終了した場合のみトリガー機能を使用する設定を追加したユーザインターフェースを示す図である。

【図 15】

本発明の第 3 の実施形態にかかるログ取得方法を実現するソフトウェア評価シ

システムの、関数／メソッドのエラー定義の内容を示した図である。

【図 16】

本発明の第3の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、図14で設定した内容に従い、ログを取得する場合の流れを示す図である。

【図 17】

本発明の第3の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、図16のステップS27で示した通常のログ取得処理の詳細の流れを示す図である。

【図 18】

本発明の第3の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、図14で設定した内容に従い、ログを取得する場合の流れを示す図である。

【図 19】

本発明の第4の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、インターフェースとメソッドのツリー表示を行うユーザインターフェースを示す図である。

【図 20】

本発明の第4の実施形態にかかるソフトウェア評価システムの、図19のようにログ取得対象を選択した場合の、ログを取得する際の処理の流れを示す図である。

【図 21】

本発明の第5の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、インターフェースとメソッドのツリー表示を行うユーザインターフェースを示す図である。

【図 22】

本発明の第5の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、図21のようにログ取得対象を選択した場合の、ログを取得する際の処理の流れを示す図である。

【図 2 3】

本発明の第 6 の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、日付毎にログを分割保存する場合の処理の流れを示す図である。

【図 2 4】

本発明の第 7 の実施形態にかかるログ取得方法を実現するソフトウェア評価システムの、サイズ毎もしくは個数毎にログを分割保存する場合の処理の流れを示す図である。

【図 2 5】

本発明の第 8 の実施形態にかかるログ取得方法を実現するソフトウェア評価システムにおいて、取得したログを一定数だけメモリに保存する場合のメモリの概要図である。

【図 2 6】

本発明の第 8 の実施形態にかかるログ取得方法を実現するソフトウェア評価システムにおいて、取得したログを一定数だけメモリに保存する場合のログ取得の流れを示す図である。

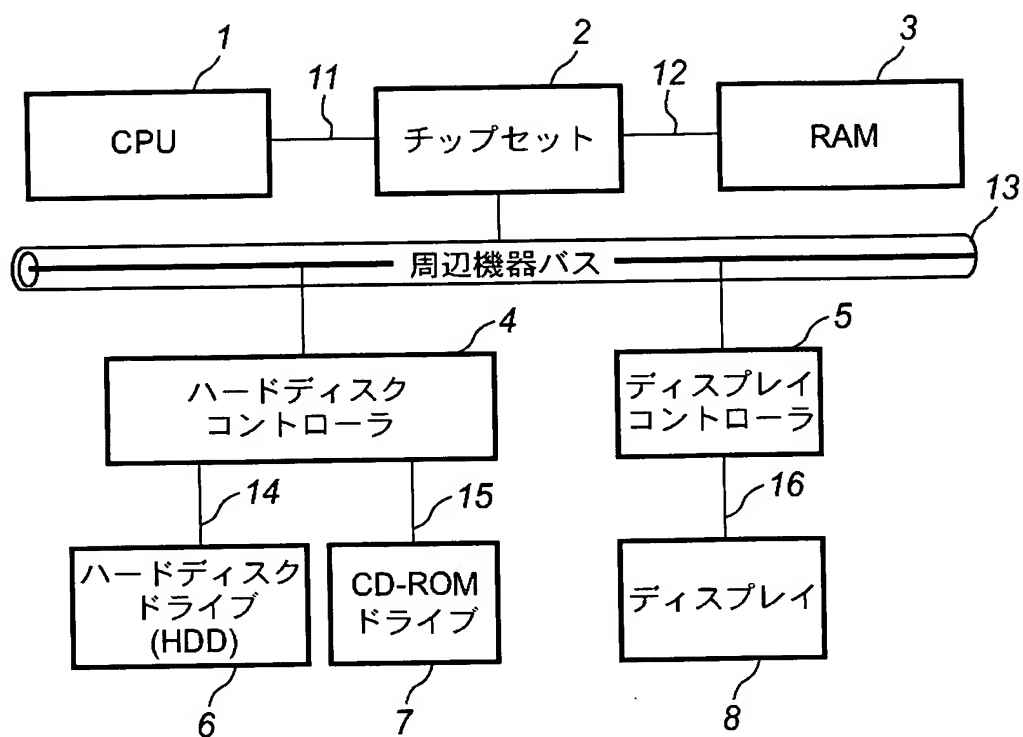
【符号の説明】

- 1：CPU
- 2：チップセット
- 3：RAM
- 4：ハードディスクコントローラ
- 5：ディスプレイコントローラ
- 6：ハードディスクドライブ
- 7：CD-ROMドライブ
- 8：ディスプレイ
- 11：CPUとチップセットを繋ぐ信号線
- 12：チップセットとRAMを繋ぐ信号線
- 13：チップセットと各種周辺機器とを繋ぐ周辺機器バス
- 14：ハードディスクコントローラとハードディスクドライブを繋ぐ信号線
- 15：ハードディスクコントローラとCD-ROMドライブを繋ぐ信号線

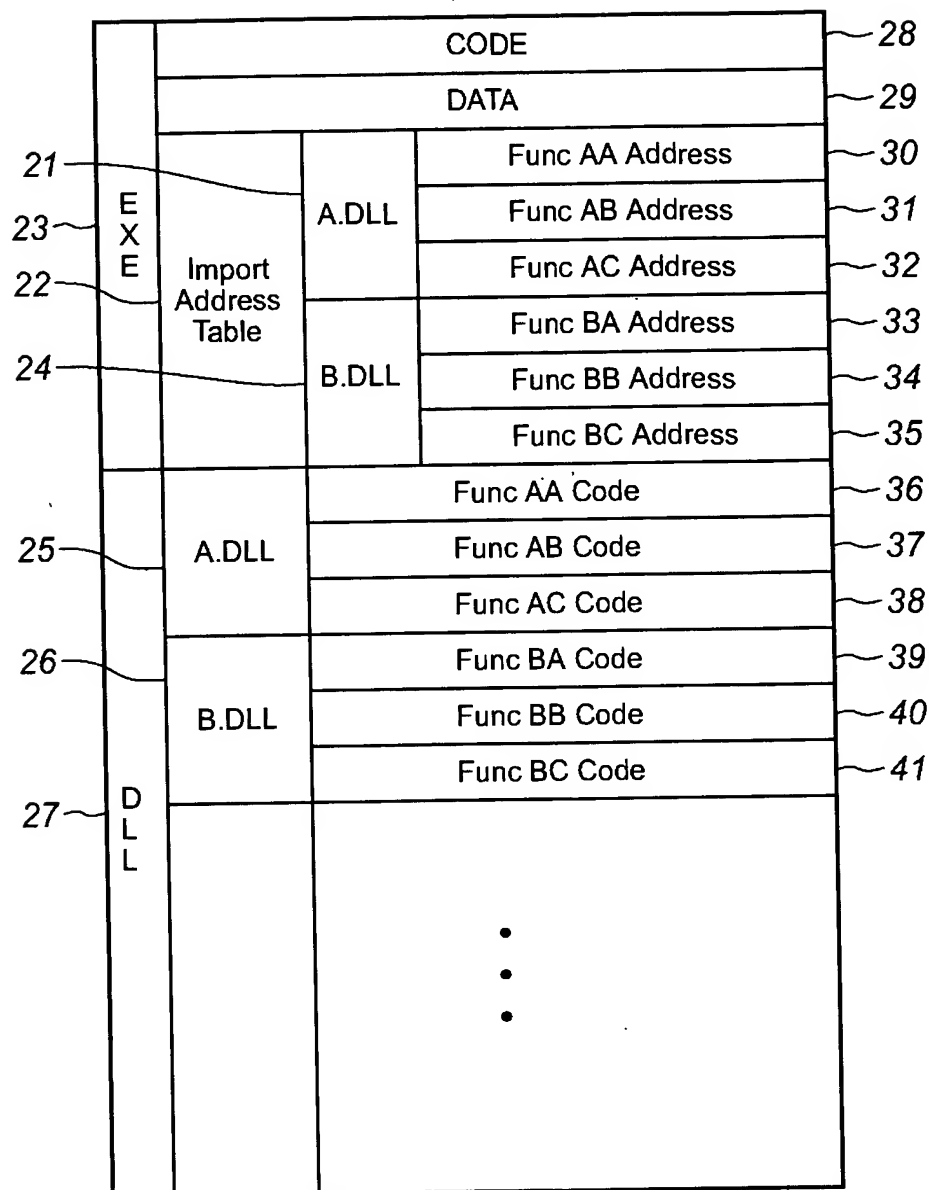
1 6 : ディスプレイコントローラとディスプレイを繋ぐ信号線

【書類名】 図面

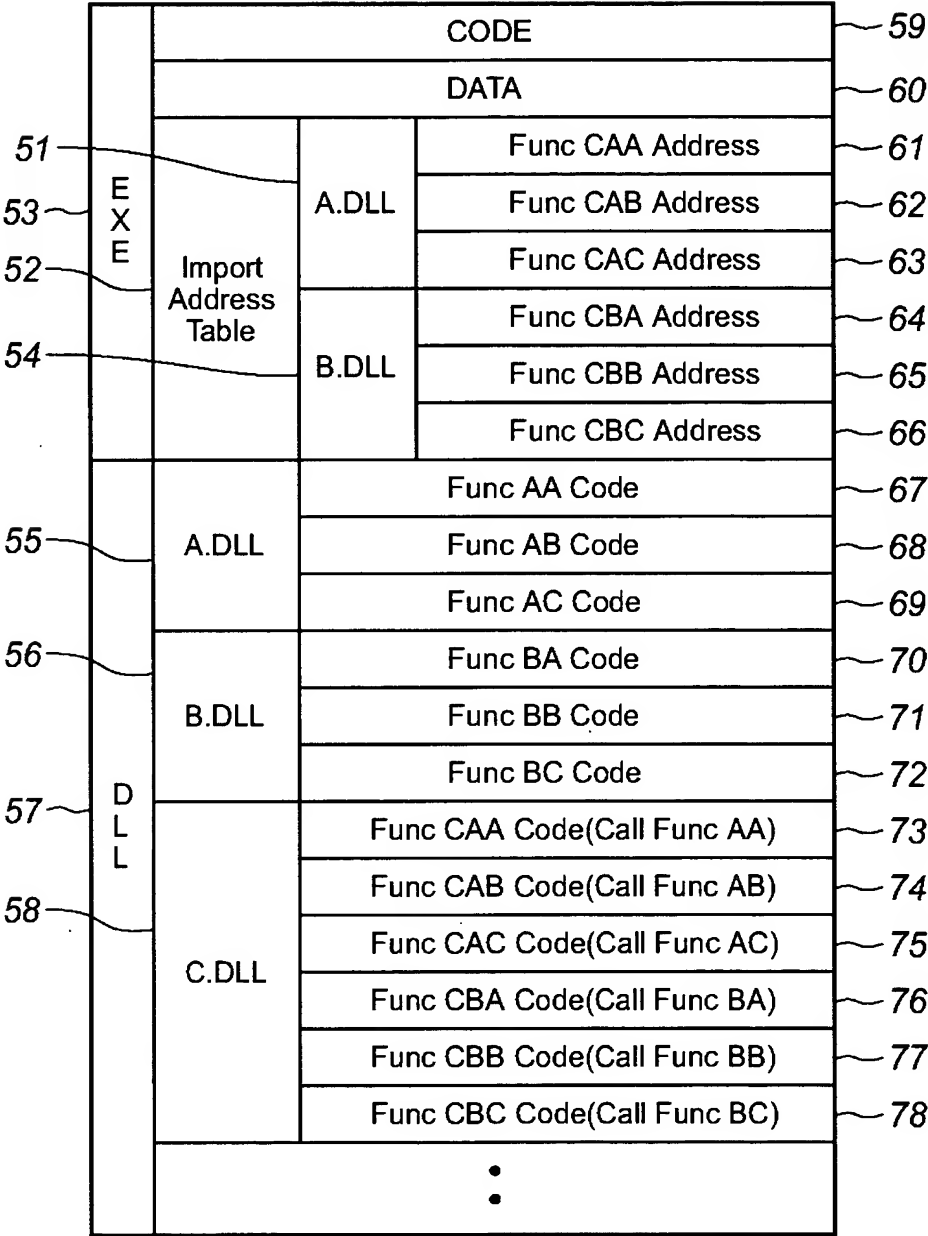
【図 1】



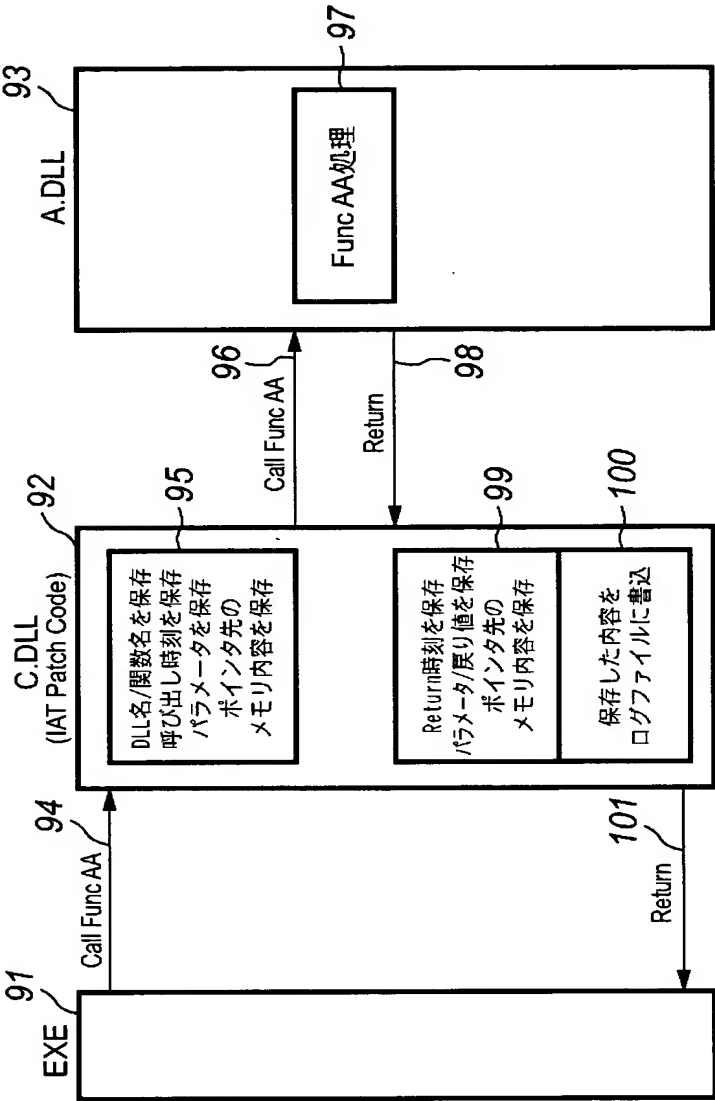
【図 2】



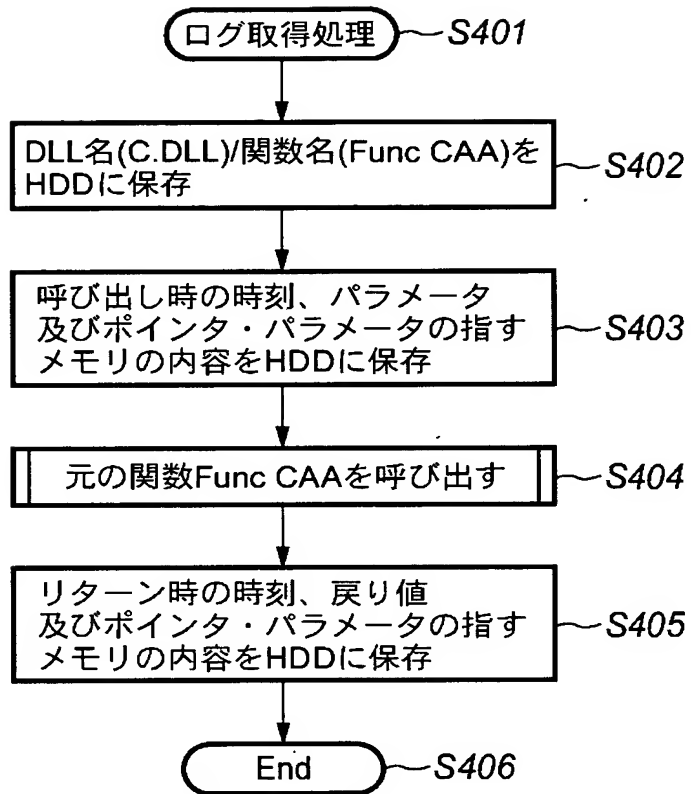
【図 3】



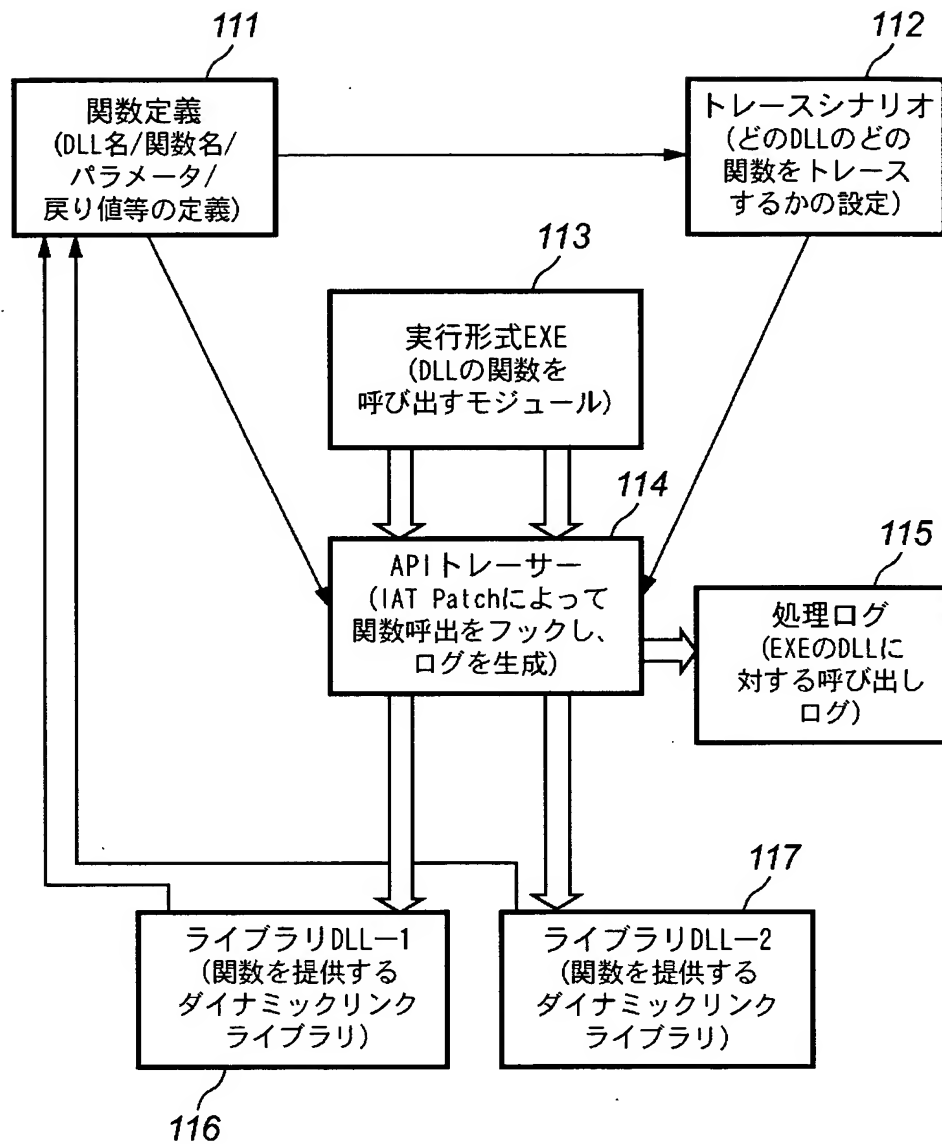
【図 4 A】



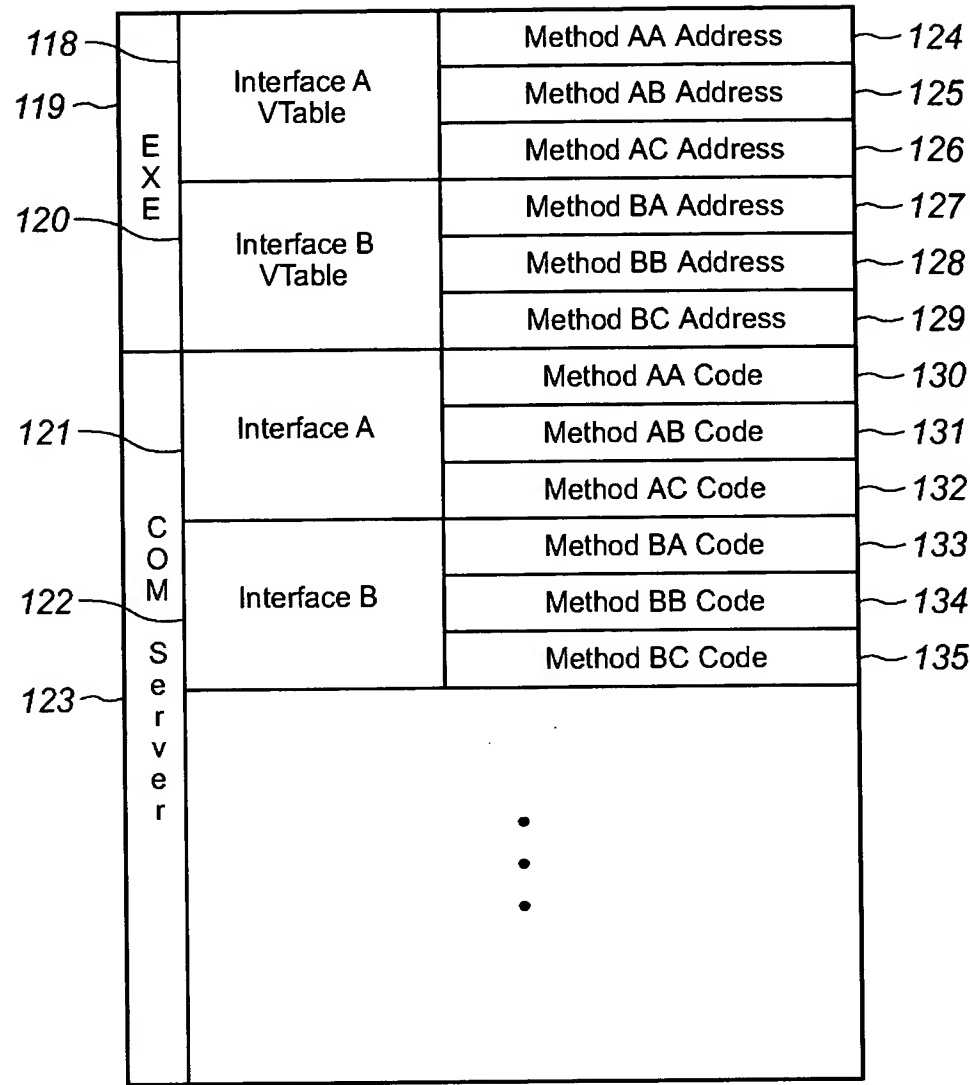
【図 4 B】



【図 5】



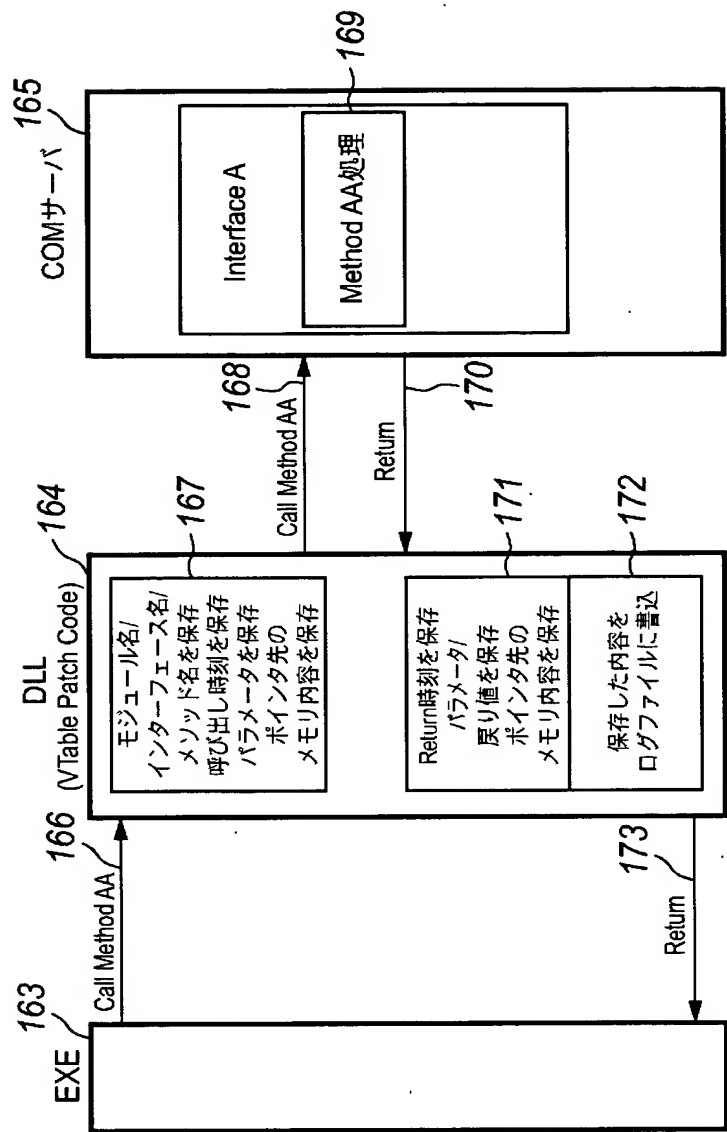
【図 6】



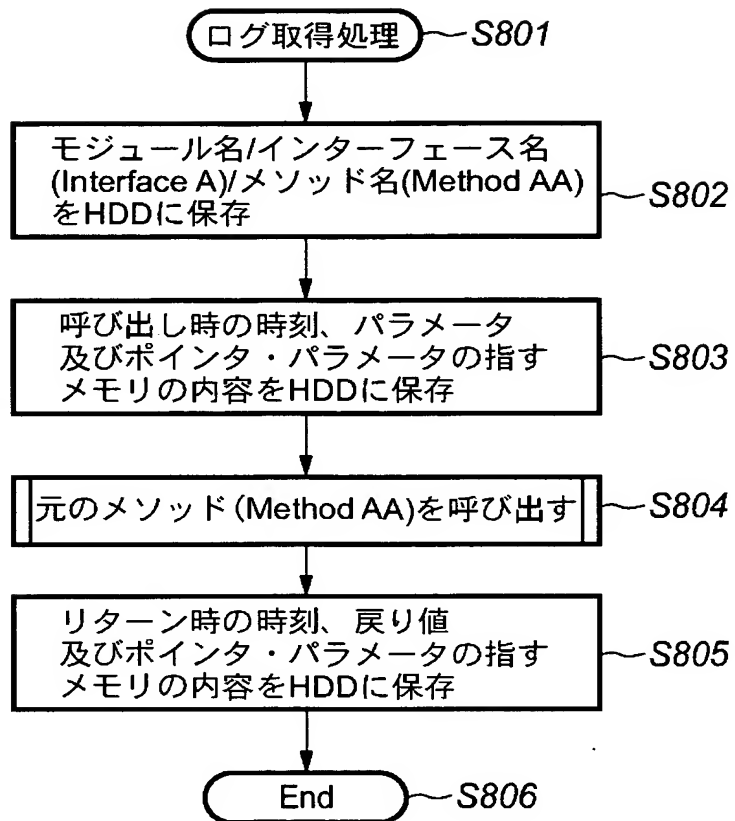
【図 7】

136	E X E	Interface A' VTable	Method A' A Address	145
137			Method A' B Address	146
			Method A' C Address	147
138		Interface B' VTable	Method B' A Address	148
			Method B' B Address	149
			Method B' C Address	150
139	C O M S e r v e r	Interface A	Method AA Code	151
			Method AB Code	152
			Method AC Code	153
141		Interface B	Method BA Code	154
			Method BB Code	155
			Method BC Code	156
140			• •	
142	D L L	Interface A'	Method A' A Code (Call Method AA)	157
			Method A' B Code (Call Method AB)	158
			Method A' C Code (Call Method AC)	159
143		Interface B'	Method B' A Code (Call Method BA)	160
			Method B' B Code (Call Method BB)	161
144			Method B' C Code (Call Method BC)	162

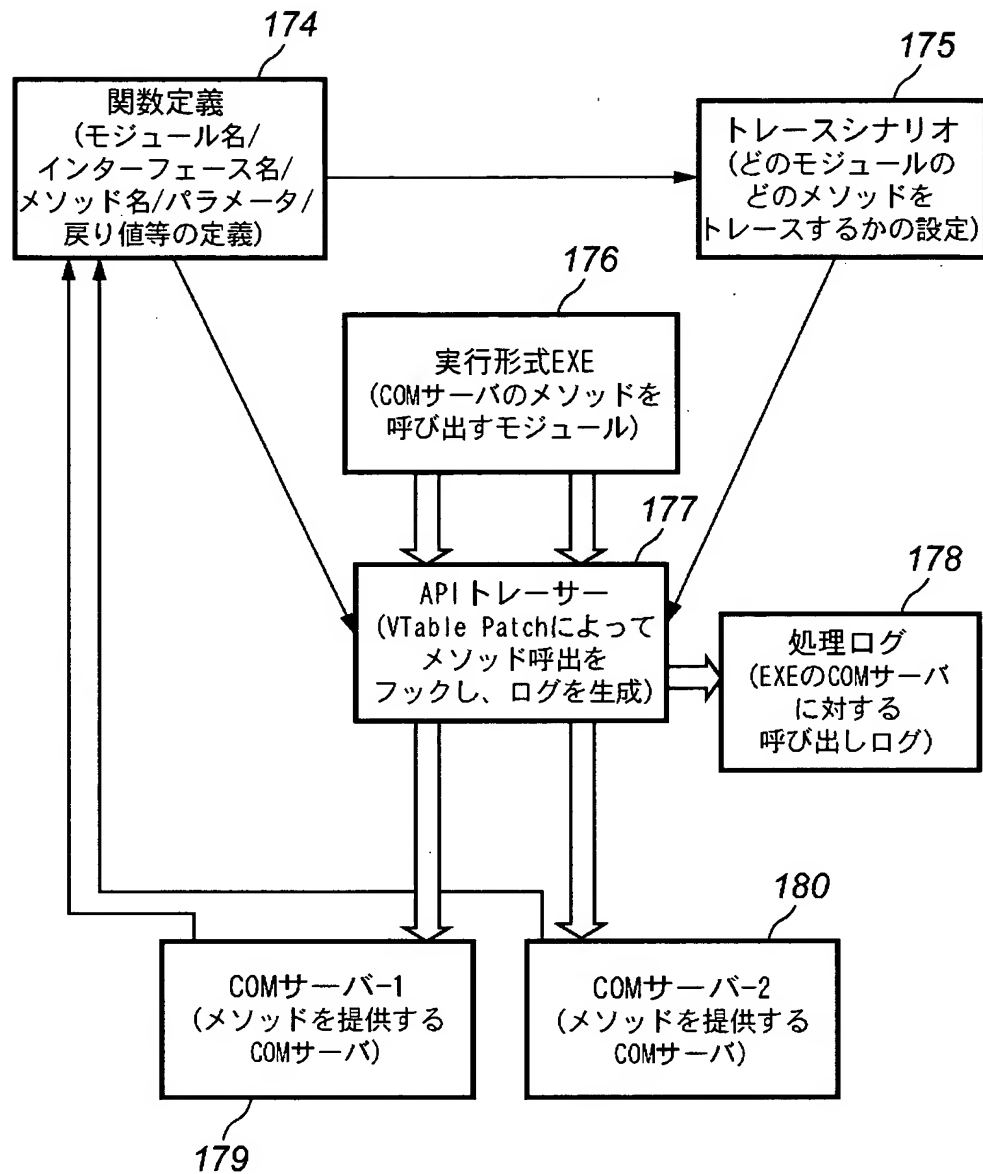
【図8A】



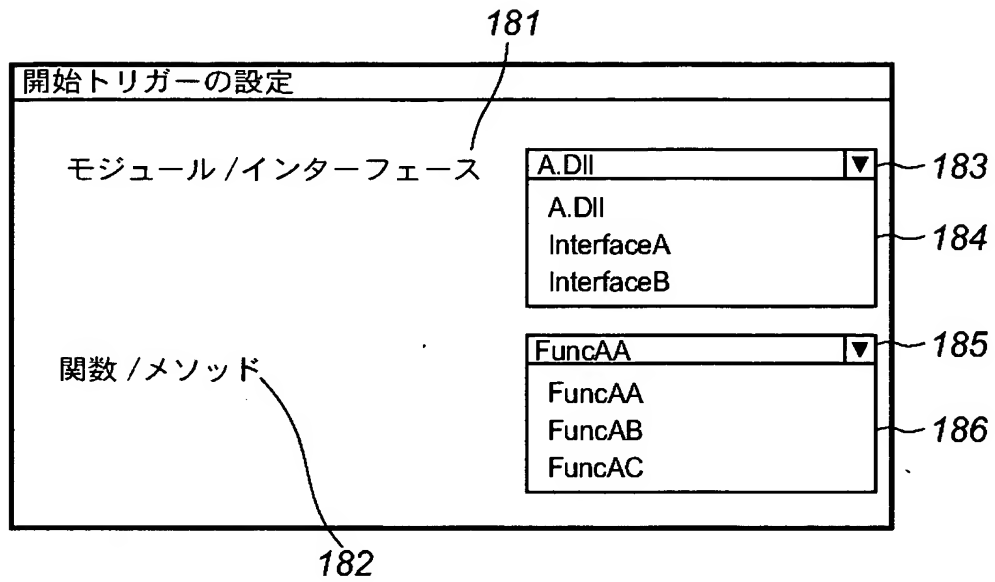
【図 8 B】



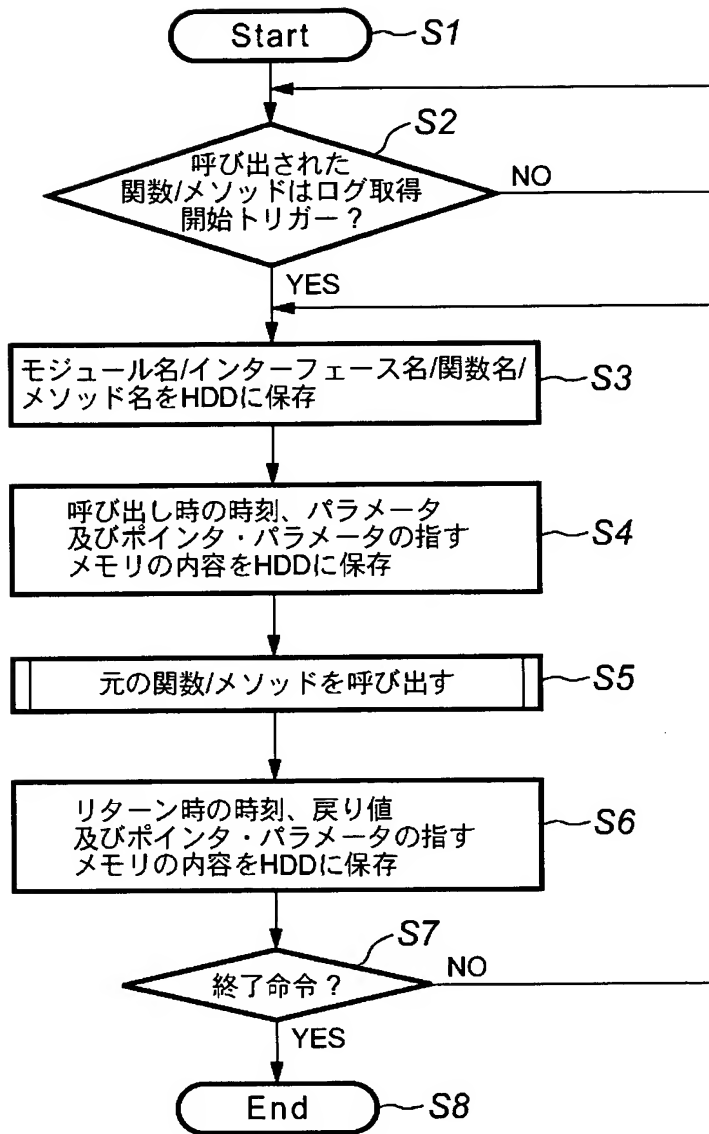
【図 9】



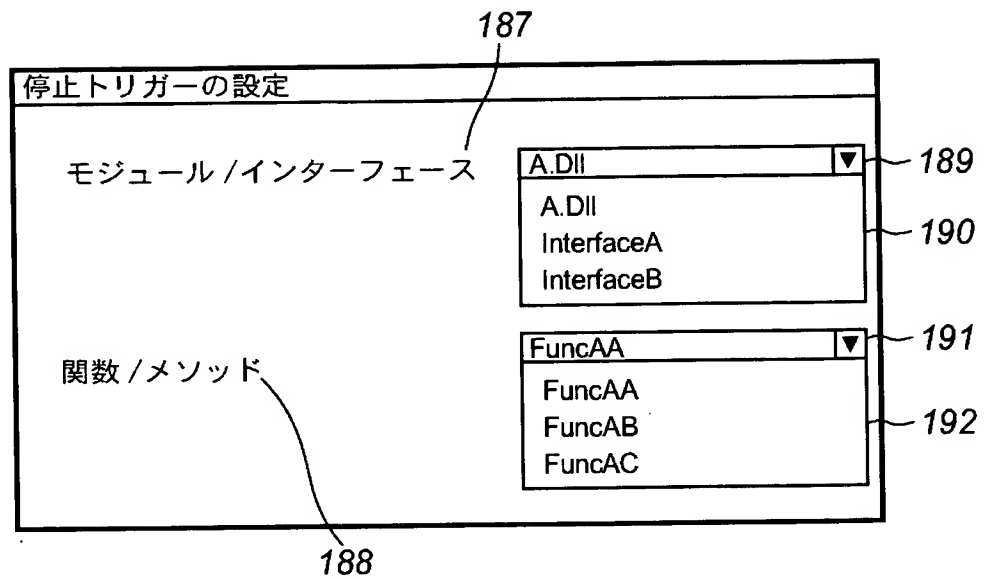
【図 10】



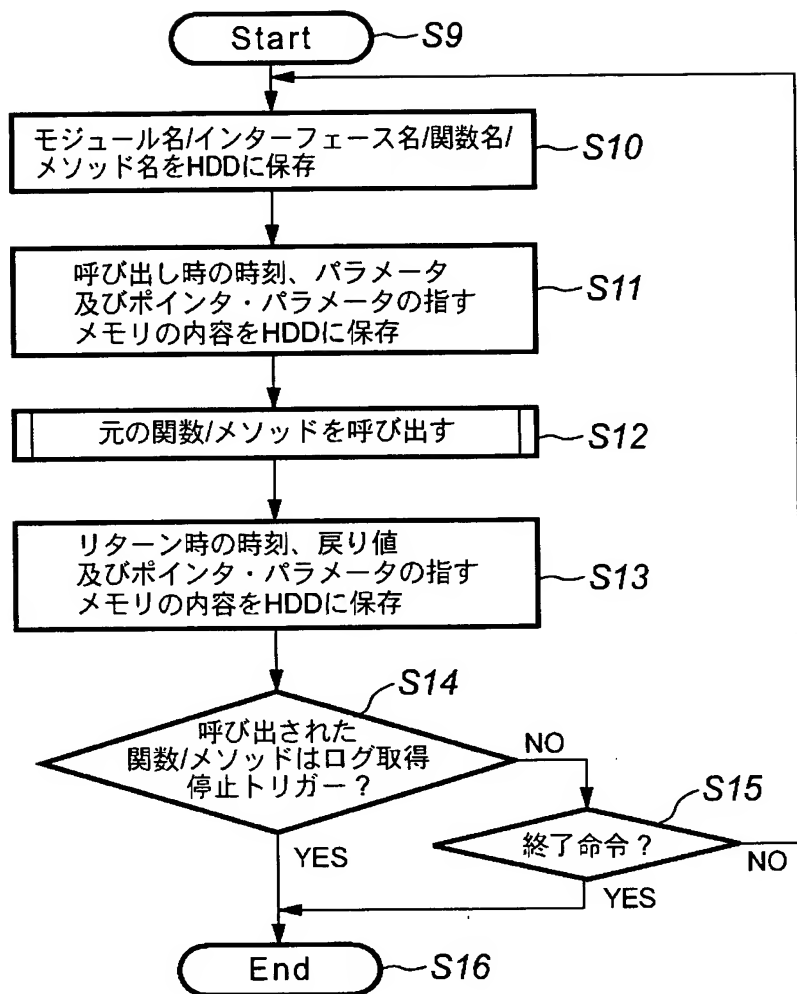
【図 11】



【図 12】



【図 13】



【図 14】

193

エラートリガーの設定

モジュール/インターフェース

関数/メソッド

194

199

195

196

197

198

A.Dll

A.Dll
InterfaceA
InterfaceB

FuncAA

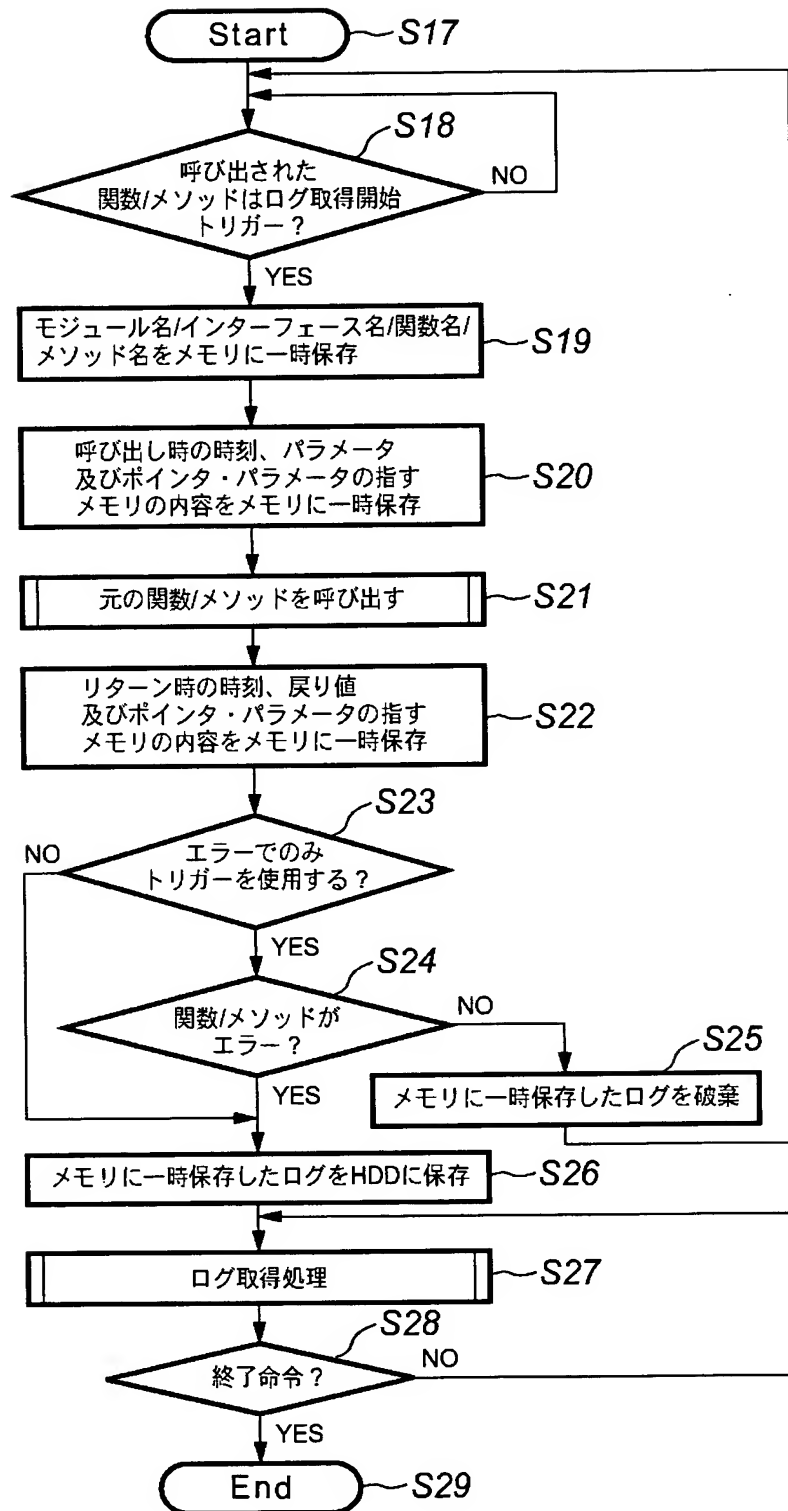
FuncAA
FuncAB
FuncAC

☒ 関数/メソッドがエラーで終了した場合のみトリガーを使用する

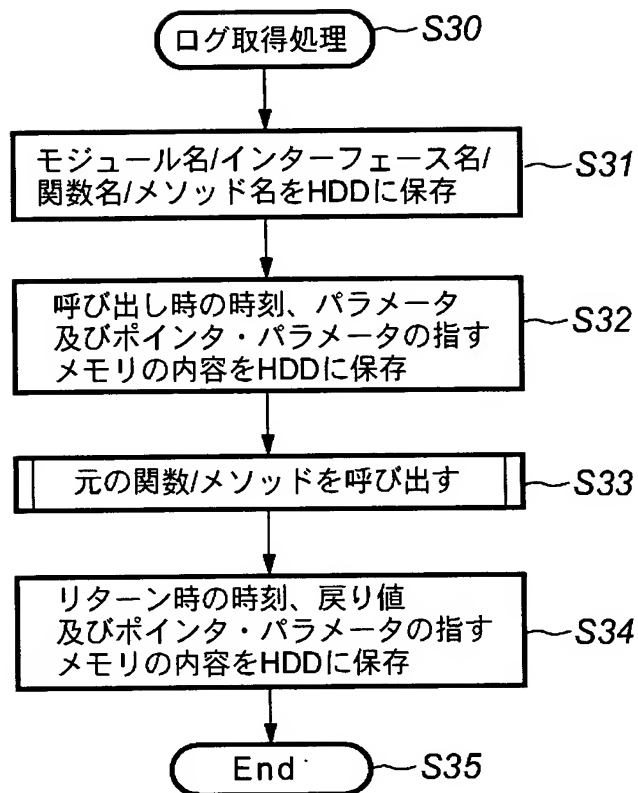
【図 15】

モジュール名:	A. DLL
関数名:	FuncAA
引数:	DWORD dwID: Err>100
戻り値:	DWORD dwRet: Err==0
モジュール名:	A. DLL
関数名:	FuncAB
引数:	DWORD dwHandle: Err==0
戻り値:	int nRet: Err<=-1
モジュール名:	B. DLL
インターフェース名:	InterfaceA
メソッド名:	MethodAA
引数:	DWORD dwID: Err>100
戻り値:	DWORD dwHandle: Err==0
モジュール名:	B. DLL
インターフェース名:	InterfaceA
メソッド名:	MethodAB
引数:	DWORD dwID: Err<=0
戻り値:	DWORD dwRet: Err!=0
モジュール名:	B. DLL
インターフェース名:	InterfaceB
メソッド名:	MethodBA
引数:	DWORD dwID: Err>=0
戻り値:	DWORD dwRet: Err!=0

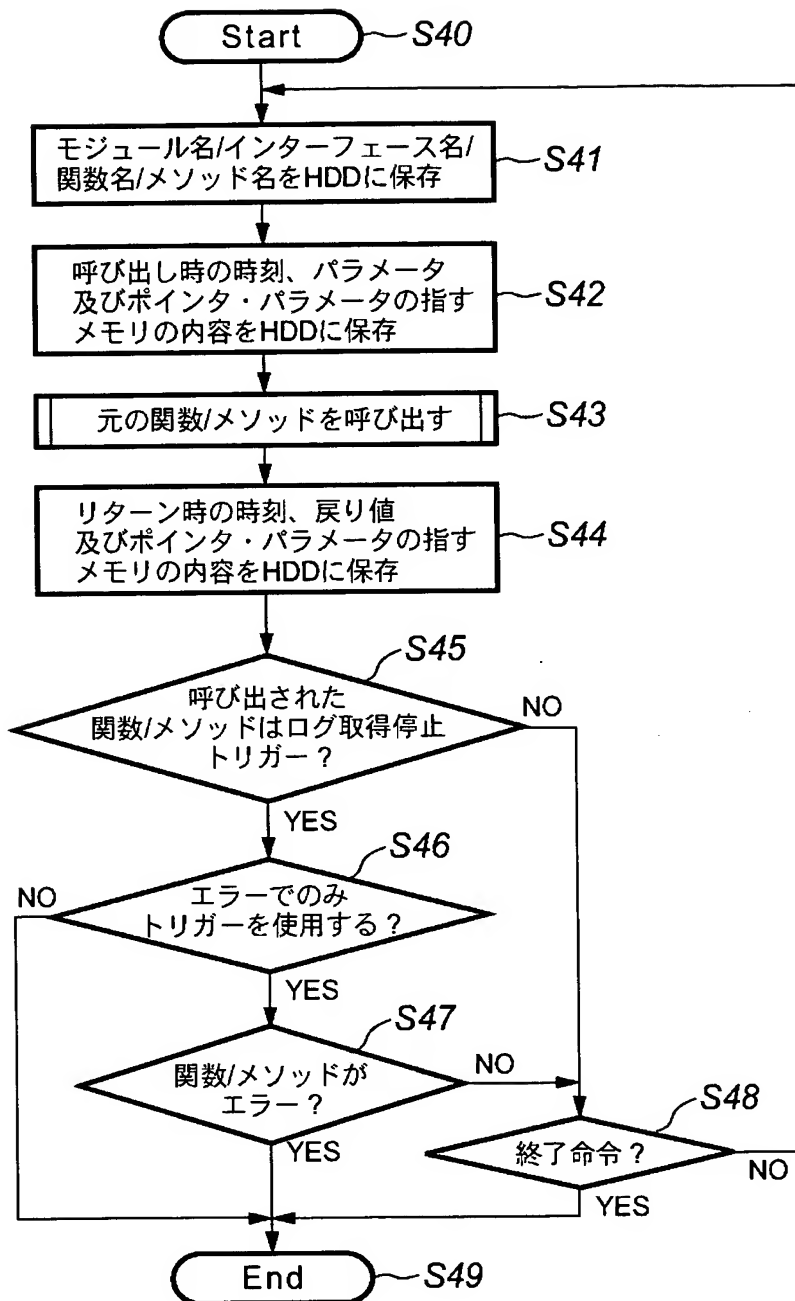
【図 16】



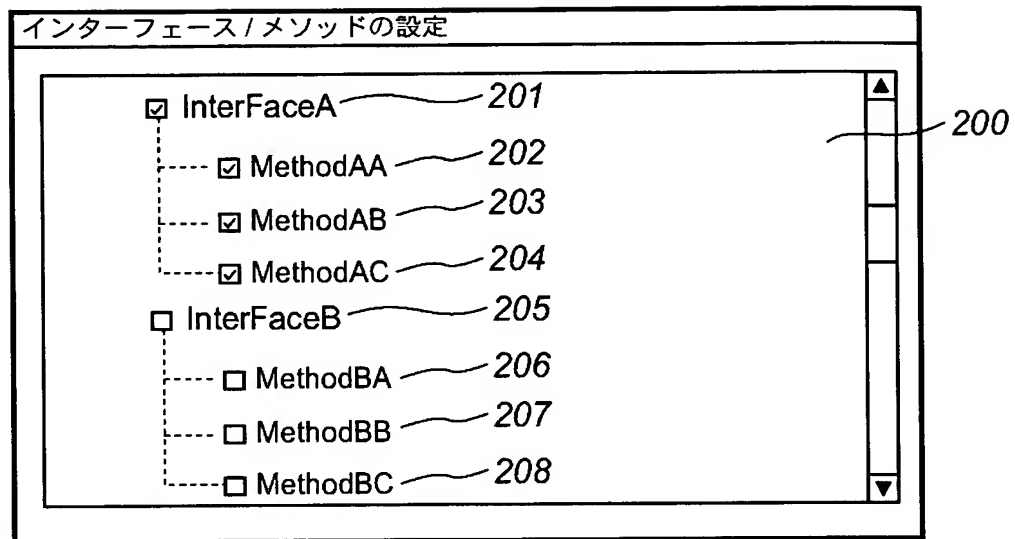
【図 17】



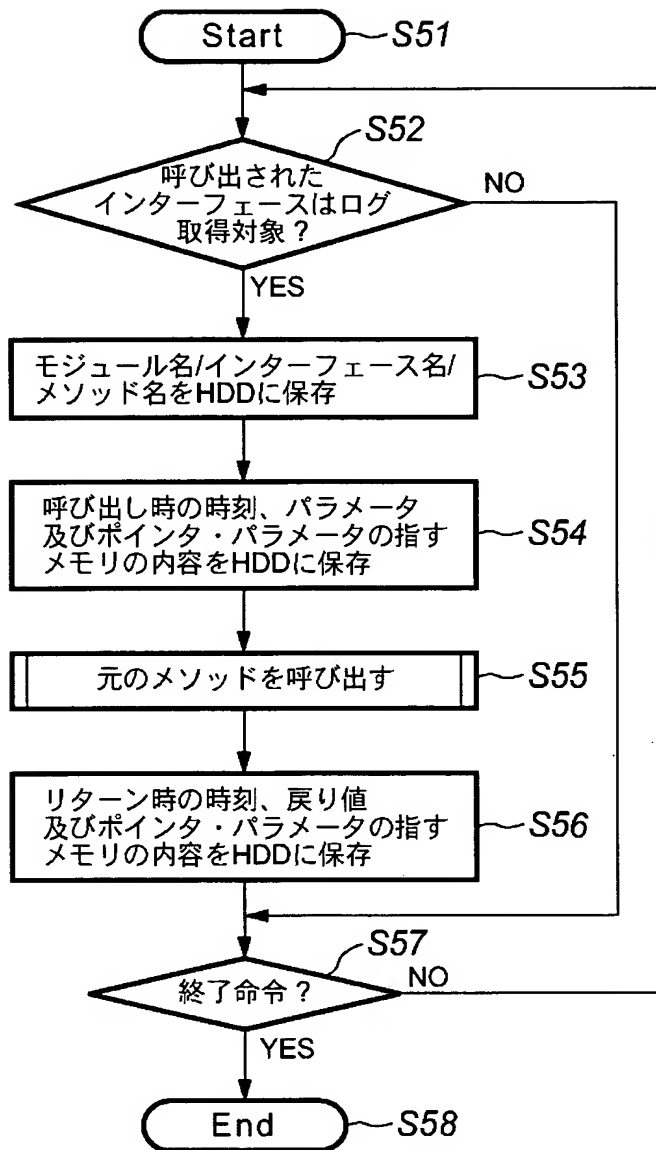
【図 18】



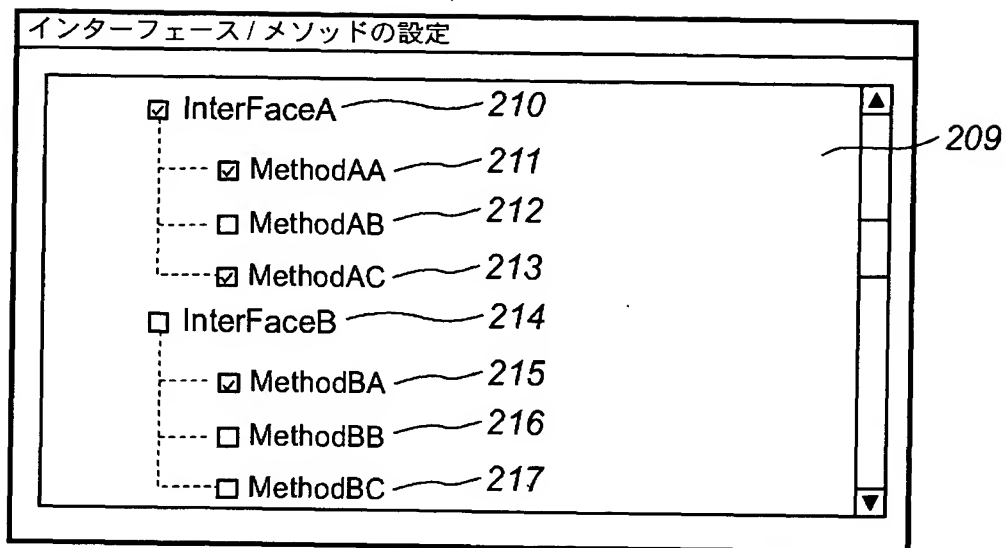
【図 19】



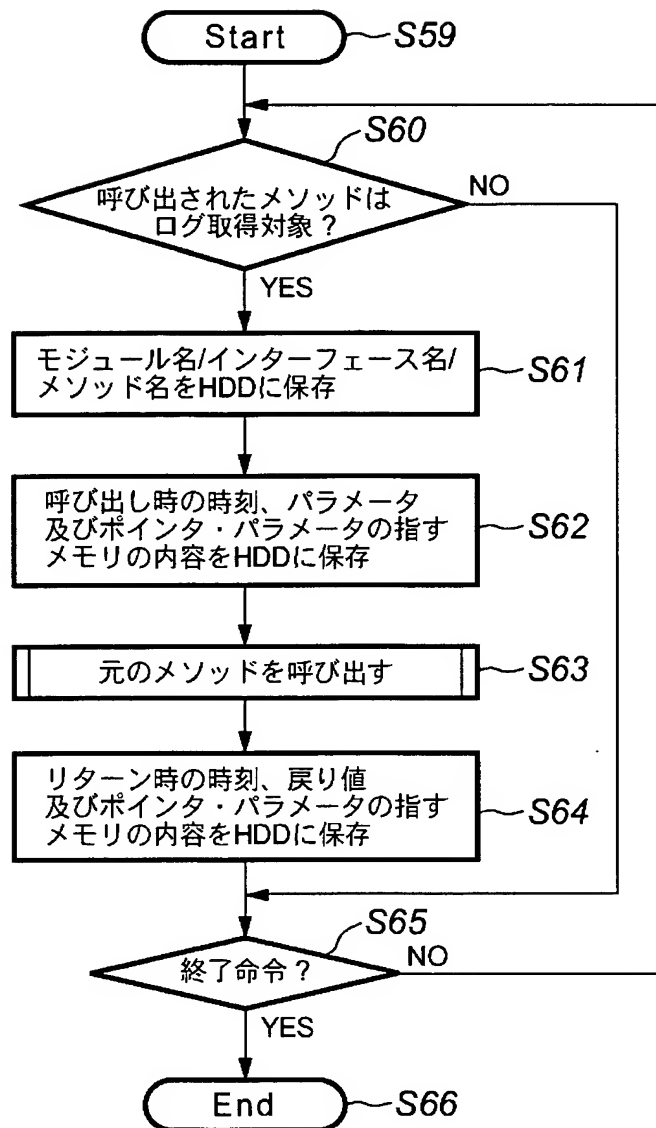
【図 20】



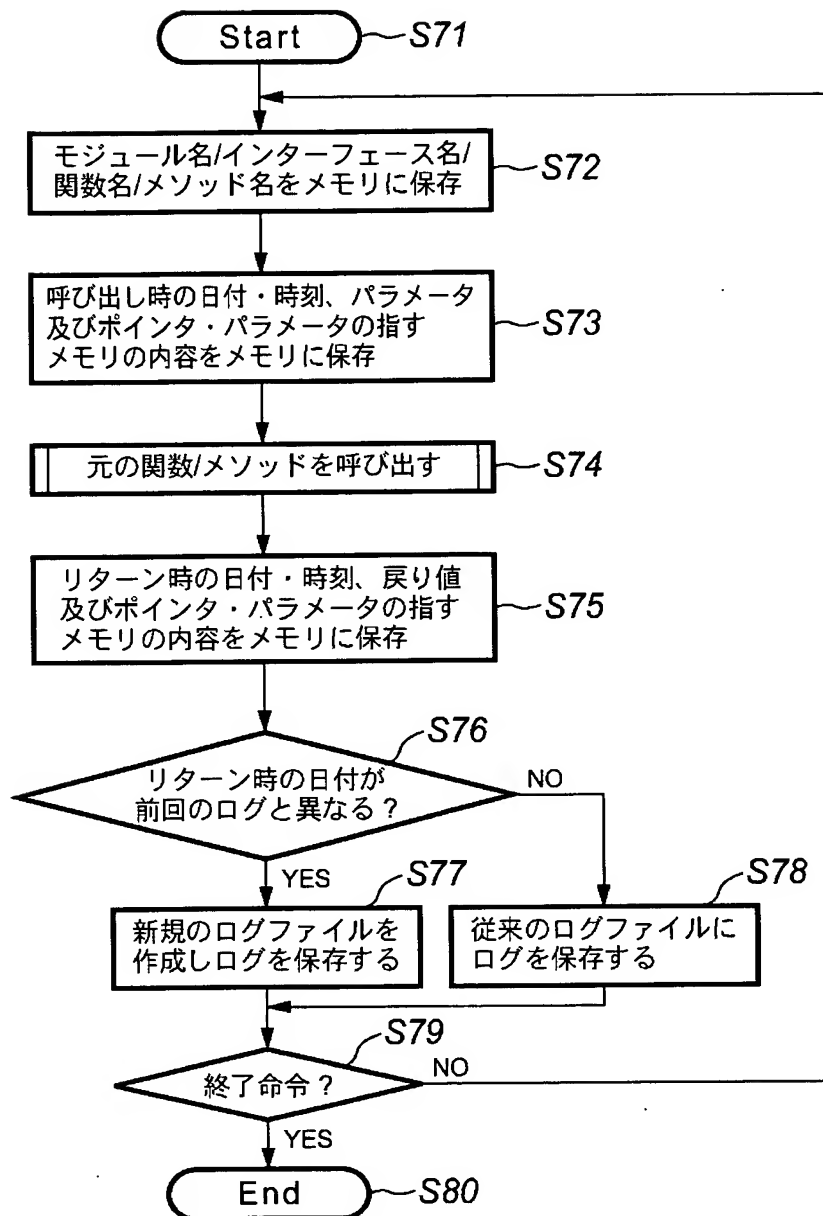
【図 21】



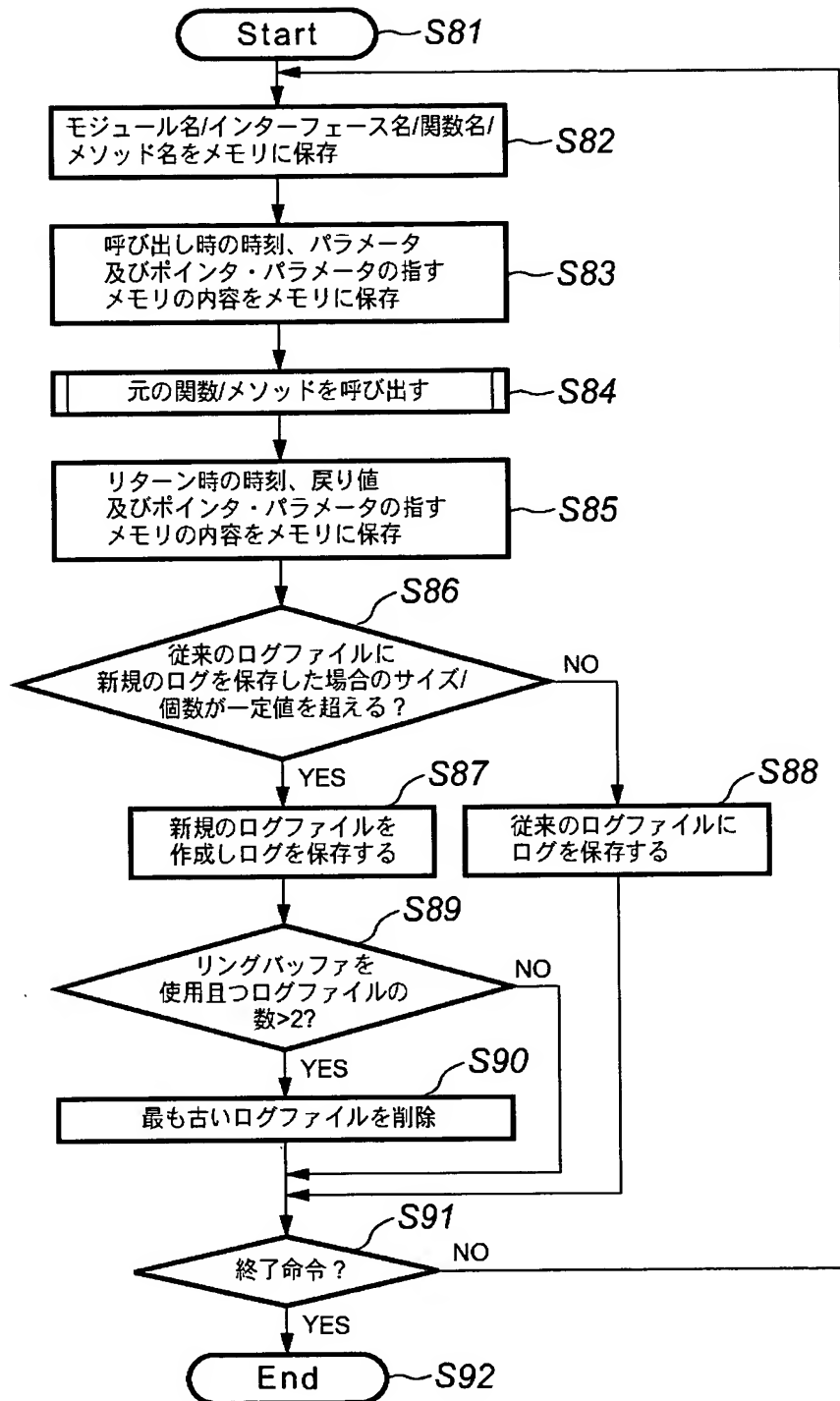
【図 22】



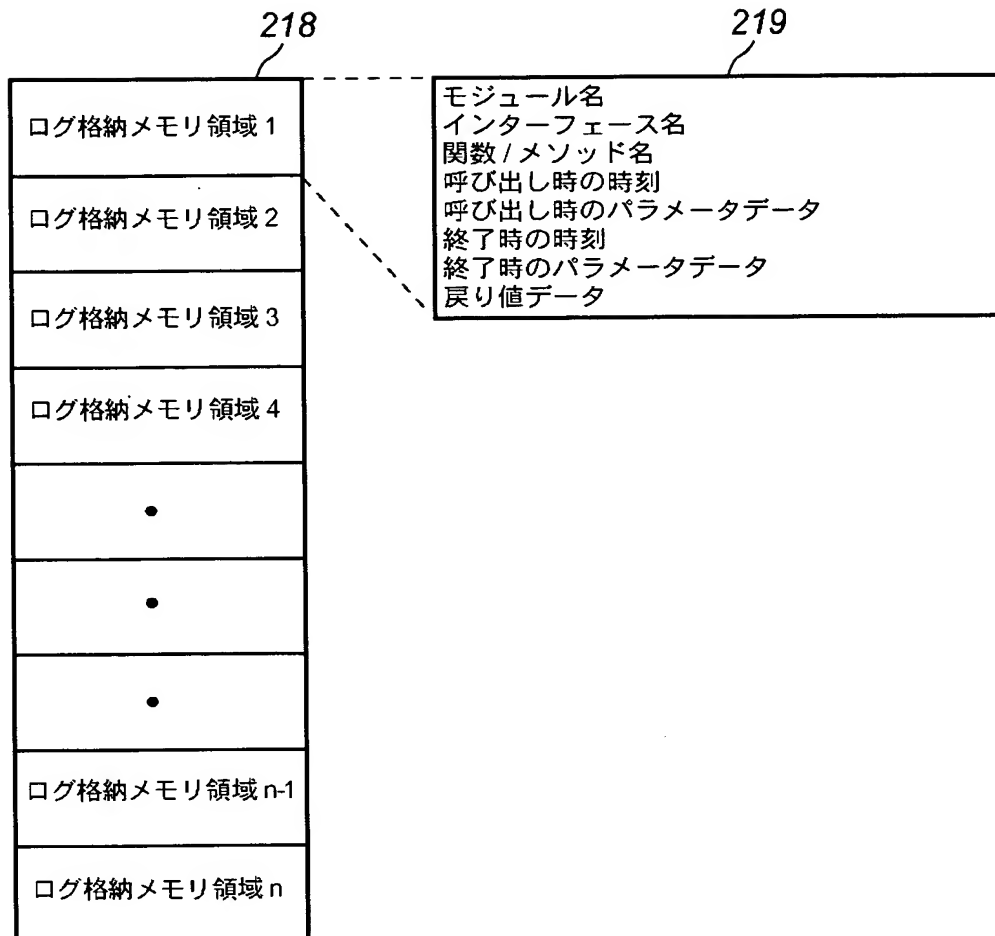
【図 23】



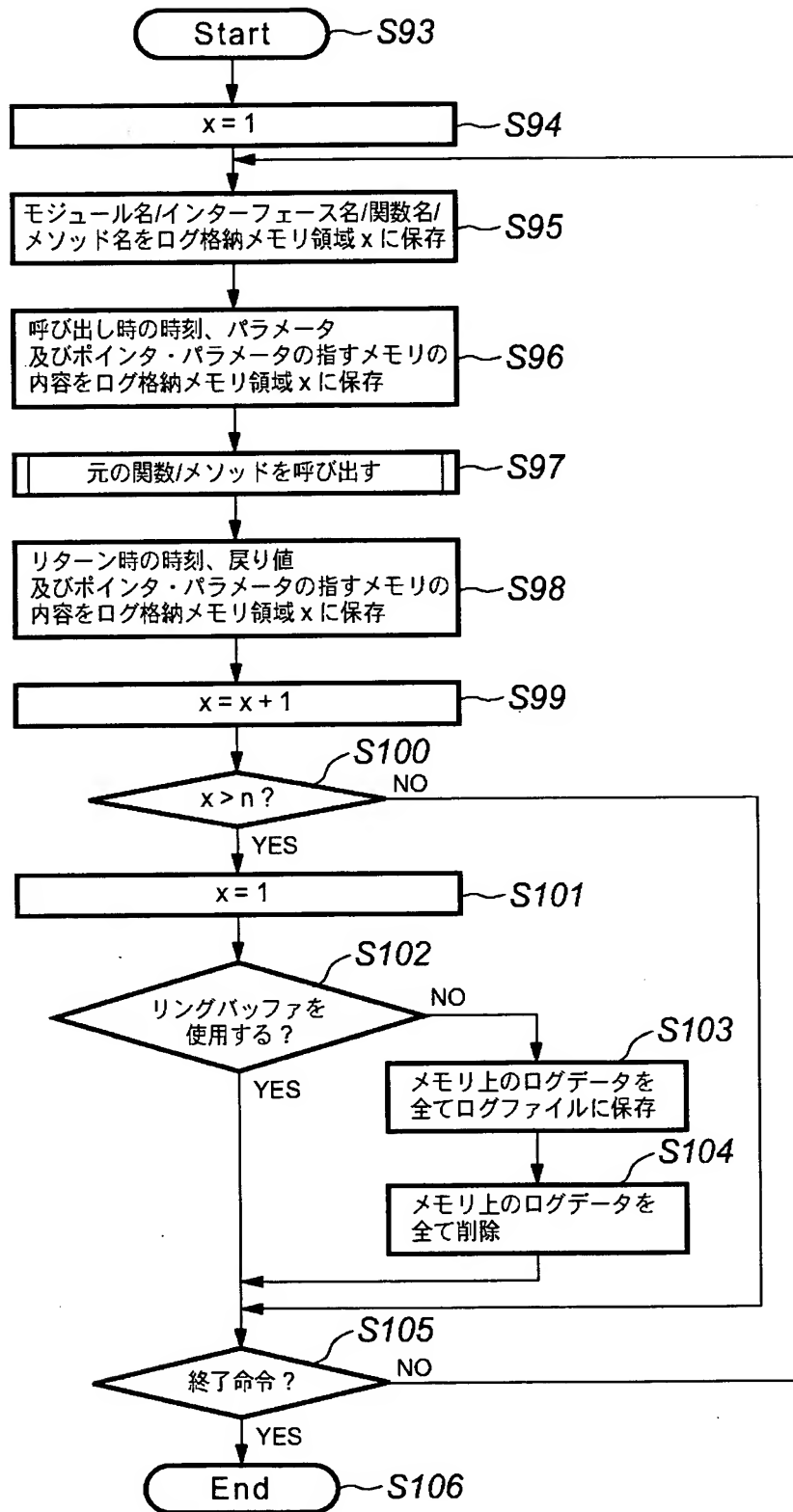
【図 24】



【図 25】



【図 26】



【書類名】 要約書

【要約】

【課題】 ソフトウェアの処理ログを容易に取得でき、かつ、バグの解析のための工数を削減することが可能なログ取得方法を提供する。

【解決手段】 関数 (FuncAA) を備えるプログラム 9 1 において、実行中のログを取得するログ取得方法であって、ロードされた関数 (FuncAA) のアドレスを、ログ取得のための関数 (9 2) のアドレスに書き換える工程と、関数 (FuncAA) を選択する工程と、を備え、ログ取得のための関数 (9 2) は、関数 (FuncAA) のアドレスを呼び出し (9 6)、該所定の処理を実行させ (9 7)、受け取った実行結果 (9 8) をプログラム 9 1 に受け渡す工程 (1 0 1) と、前記選択された関数 (FuncAA) のアドレスを呼び出す際の所定の情報を記録する工程 (9 5、1 0 0) と、前記選択された関数 (FuncAA) の前記実行結果を受け取った際の所定の情報を記録する工程 (9 9、1 0 0) とを備える。

【選択図】 図 4 A

特願 2002-191129

出 願 人 履 歴 情 報

識別番号

[000001007]

1. 変更年月日

1990年 8月30日

[変更理由]

新規登録

住 所

東京都大田区下丸子3丁目30番2号

氏 名

キヤノン株式会社